Nominal record linkage: the development of computer strategies to achieve the family-based record linkage of nineteenth century demographic data

Thesis

NOMINAL RECORD LINKAGE:
THE DEVELOPMENT OF COMPUTER STRATEGIES TO ACHIEVE THE FAMILY-BASED RECORD LINKAGE OF NINETEENTH CENTURY DEMOGRAPHIC DATA

by

JOHN ANTHONY WELFORD, B.Sc.

A Thesis Submitted

for

the Ph.D. Degree

Volume 1

of 2

Applied Historical Studies

The Faculty of Social Sciences

The Open University

July 1989

CONTENTS

VOLUME 1

FIGURES

MAP

TABLES

PREFACE

This thesis was originally submitted for examination in March 1983. Following the result of the viva in October of that year an appeal was lodged, and the subsequent proceedings lasted for almost four years. In October 1987 formal notification was made that the thesis could be revised and resubmitted.

The prolonged length of the appeal proceedings has meant that the computing environment within which the research was set has developed significantly from the position in 1983. Indeed, in purely practical terms, the computing systems which were used at that time are no longer operational. The opportunity for making modifications and refinements to the record linkage system, and for incorporating additional primary source materials (even were sufficient human resources available), has therefore been removed.

Under these circumstances, the record linkage strategies described in the revised thesis are precisely the same as those presented in the 1983 submission. For this reason and because of the extensive delays in carrying out the appeal proceedings it has not seemed appropriate to provide a full review of developments in the record linkage field beyond this date.

Reference has, however, been made to the subsequent, crucial impact of the findings of the present research on the progress of the later phases of the 1851 Census National Sample Project, which I co-directed with Professor Michael Anderson at the University of Edinburgh. The entire conceptual and strategic approach to the organisation of family information in this project grew directly from perceptions which were central to the present research. Reference has also been made to the influence of the present research on the development of the SASPAC package, a computing system for handling the 1981 Population Census Small Area Statistics data for Great Britain. I was the chief systems designer of SASPAC, and the design and implementation methods which were adopted in this development drew heavily on the experience gained from the present research.

Finally, the opportunity has been taken (in the new Section 10.2), to present the findings of some fresh analyses of 1851 household census data which serve to confirm the validity of the linkage strategies which have been developed.

ACKNOWLEDGEMENTS

I should like to express my indebtedness to my research supervisors, Professor Michael Anderson and Professor Michael Drake, for their sustained guidance and encouragement during my work on this thesis. I am also grateful to the many others who have provided helpful advice at various stages. Special thanks are due to the staff of the Edinburgh University Computing Service (formerly Edinburgh Regional Computing Centre), and, in particular, to the then Database Consultant, Dr Geoff Stacey, for the invaluable assistance which was given with the use of the IDMS database management system and other facilities. I should also like to add my thanks to Mr Ray Thomas of the Open University for his more recent supervisory involvement, following the retirement of Professor Drake. While those mentioned above have guided and stimulated my efforts I must claim the ultimate responsibility for the degree to which I have, or have not, heeded their advice. Except where explicitly acknowledged in the text, the ideas presented in this thesis represent my own original research. No part of the thesis has previously been published elsewhere.

Grateful acknowledgement must also be made to the Economic and Social Research Council for the award of a three-year research fellowship (1976-9), during which the bulk of the computer system development work was completed.

ABBREVIATIONS AND SYMBOLS

The following abbreviations have been used, mainly in the bibliography and notes:

ANSI	American National Standards Institute
BURISA	British Urban and Regional Information Systems Association
CODASYL	Committee on Data Systems Languages
CSO	Central Statistical Office
DBMS	database management system
DCRO	Durham County Record Office
EMAS	Edinburgh Multi-Access System
ERCC	Edinburgh Regional Computing Centre
ESRC	Economic and Social Research Council (formerly SSRC)
HMSO	Her Majesty's Stationery Office
HO	Home Office
IBM	International Business Machines
ICL	International Computers Limited
IDMS	Integrated Database Management System
IMS	Information Management System
ISO	International Organisation for Standardization
LAMIS	Local Authority Management Information System
LAMSAC	Local Authorities Management Services and Computer Committee
NUMAC	Northumbrian Universities Multiple Access Computer
OPCS	Office of Population Censuses and Surveys
PP	Parliamentary Papers
PRO	Public Record Office
SIR	Scientific Information Retrieval
SPSS	Statistical Package for the Social Sciences
SQL	Structured Query Language
SSRC	Social Science Research Council
VME	Virtual Machine Environment

The following abbreviations and symbols appear in the computer-produced listings and also frequently in the text:

ANG	Anglican
(B)	base zone
BACH	bachelor
BAN	banns
BAP	baptism
BNS	banns
BUR	burial

Bnnnnn	unique birth record identifier, e.g. B00305
B1	format prefix tag for post-1812 baptism records
B2	format prefix tag for 1798-1812 baptism records
C	continuation character for all command and data input
CE	census
Cnnnnn	unique census record identifier, e.g. C00012
C5	format prefix tag for 1851 census records
C6	format prefix tag for 1861 census records
C7	format prefix tag for 1871 census records
D	daughter (and also 'days' in age specification in data input formats)
DAUR	daughter
DD	prefix tag for each command submitted to the Directory Prime Subsystem
Dnnnnn	unique death record identifier, e.g. D00063
D3	format prefix tag for post-1812 burial records
F	female
FA	full age (i.e. aged 21 or over)
Fnnnnn	unique family identifier, e.g. F00108
I	illegitimate
LIC	licence
M	male (and also 'months' in age specification in data input formats)
MAR	marriage
Mnnnnn	unique marriage record identifier, e.g. M00024
M1	format prefix tag for pre-1837 marriage records
M2	format prefix tag for post-1837 marriage records
(N)	non-base zone
OTP	of this parish
(P)	peripheral zone
PP	prefix tag for each command submitted to the Record Linkage and Population Analysis Subsystems
PR	parish register
Pnnnnn	unique person identifier, e.g. P00230
S	son
SIG	signed
SPI(N)	spinster
SS	prefix tag for each command submitted to the Source Translation and Source Loading Subsystems
UNM	unmarried
W	weeks
WDR	widower

WID	widow
X	unsigned
:	prefix used to indicate the start of a user command
@@@@	missing information
+-	plus or minus. For example, '+- 183D' indicates that a date is being specified as accurate to the nearest half-year (i.e. 183 days)

Abbreviated forms are occasionally used, for convenience, when making repeated reference to particular COBOL program modules. For example, 'DW' is used as an abbreviation for 'DWDBSEARCH' in Chapter 5. Such two-character forms are guaranteed to provide unique module identification. (The policy adopted for naming modules is described in Chapter 5, Note 8.)

Footnotes have been collected together and presented at the end of each chapter. References appear within the text in parentheses, e.g. (7).

References to the user or researcher as 'he' are of course to be interpreted as signifying both male and female.

CHAPTER 1 INTRODUCTION

1.1 THE PROBLEM OF NOMINAL RECORD LINKAGE

This thesis is grounded in the interdisciplinary area between computing and historical demography. Its overriding purpose is to investigate the extent to which computing technology can assist the historical demographer to exploit fully the various historical source materials which are at his disposal. The focus of this investigation is the problem of nominal record linkage, a problem which has received considerable attention (1), but which has hitherto not been significantly resolved.

Dr E A Wrigley, a historical demographer with extensive experience in this field, has described nominal record linkage as (1973, 1): 'the process by which items of information about a particular named individual are associated with each other into a coherent whole in accordance with certain rules.' According to this description, therefore, nominal record linkage can be said to take place, for example, when a genealogist searches through a parish register and copies out on a sheet of paper the entries for a particular individual, such as his baptism, marriage and burial.

The above description and example, however, may serve to give a crudely simplistic impression of what can turn out in reality to be an

extremely complex process. Let us examine, therefore, some of the additional dimensions of the problem which are missing from the above description.

In the first place, the expression 'a particular named individual' may give the impression that a person is always referred to in the same way in each of his records. This is manifestly not the case, and one needs to recognise, for example, that 'Henry Fawcet' and 'Harry Forsit' may, in fact, be references to the same person. The problem is exacerbated for married women. Here one needs to recognise, for example, that 'Ann Murgatroyd' (maiden name) and 'Hannah Stephenson' (married name) may be references to the same person. Put simply, one has to recognise that for each person there can be many names, and to achieve record linkage one must devise some method of making the connection between the disparate variants.

A second, important problem which is missing from the above description of nominal record linkage concerns the non-uniqueness of people's names. If, for example, one observes in a parish register that there are three references to 'Henry Fawcet' then it is possible that they will all refer to the same person: but it is also possible that they may refer to two or even three people called 'Henry Fawcet'. This problem of non-uniqueness, simple as it is to comprehend, can be extremely difficult to resolve, particularly as there can never be complete certainty that one has reached the 'correct' decision in any linkage situation.

Thus far I have identified additional, significant dimensions of the problem of record linkage which, even if they are difficult to resolve, are at least conceptually simple to understand. I shall now address a problem which is more difficult to grasp conceptually. Implicit in the description provided by Wrigley is the assumption that record linkage is an operation which is person-based, i.e. its purpose is to draw together information for individuals, but individuals who are essentially to be considered in isolation from each other. The main outcome from such linkage will therefore be the production of a personal dossier for each individual, the dossier containing information about the person's life events, such as baptism, marriage and burial.

It is possible, however, to widen the concept of nominal record linkage by considering the linkage of demographic data which contains not just 'snapshots' of events in the lives of isolated individuals, but rather descriptions of clusters of individuals in their familial and residential contexts. The most obvious example of such a data type is the household census record, in which one will typically find the demographic details of the co-residing members of a family, together often with details of one or more non-relatives, such as servants and lodgers. When subjecting such data to the processes of record linkage one can clearly choose to ignore the larger household and familial context and instead disaggregate the data and distribute it to individual personal dossiers. In the process of linkage one would thus be losing the inter-relatedness of the members of the household and much of the informational richness which is a

characteristic of this kind of data.

An alternative approach to the linkage of census data, and one which is more faithful to the implicit structural properties of the data, is to attempt to treat the unit of linkage as the co-residing family, rather than as the individual household occupant. Thus, according to this view, when linking data from the 1851 census to data from the 1861 census one should be aiming to locate and link together occurrences of the same family in the two censuses. Given that the household family membership will normally have changed during the intervening ten years between censuses, either because of the occurrence of associated life-events such as births, marriages and deaths, or through family members (children, grandchildren, uncles, etc.) moving in or moving out of the household, the task of matching and linking such disparate family groupings is conceptually and organisationally much more complex than it is for isolated individuals. In addition, it will not be adequate to represent the end product of such linkage merely by a set of personal dossiers: there should also be the production of some kind of family dossier, not only to hold the 'life events' of the family (such as the date of marriage, date of birth of first child, etc.) but also to provide connecting links to the individual family members. Finally, since each individual is a child in one family and may at some time be a parent in one or more other families, the results of record linkage should properly be represented by family dossiers which are themselves inter-connected.

This developed concept of record linkage is clearly far removed from Wrigley's description quoted in the opening paragraph, and it is in significant advance of the essentially person-based linkage concepts implicit in record linkage work restricted to parish register and other personal event records, such as poll book and trade directory entries. In order to distinguish this type of linkage from the more primitive, person-based type I shall subsequently refer to it in the thesis as 'family-based' linkage.

It is the main objective and challenge of this thesis to explore this developed, family-based concept of record linkage and to determine whether the kinds of computing facilities which are currently available have the capability to enable corresponding linkage strategies to be developed.

1.2 EARLIER WORK IN NOMINAL RECORD LINKAGE

The foremost examination of historical nominal record linkage by computer is provided in 'Identifying people in the past' (Wrigley 1973). This book contains a number of articles on the development of record linkage systems, together with a survey of the literature (Winchester 1973). The articles cover a broad range of approaches and perceptions, and there is enormous variability in the role played by the computer. In one project (Blayo 1973) French parish registers were linked manually, with the computer being used merely as an aid to

the problem of examining and resolving variations in surnames. In a quite different project at the other end of the scale (Skolnick 1973) computer programs incorporating artificial intelligence techniques were employed to select the 'best linkages' in situations where multiple solutions were possible. Such work relied on elaborate programming and considerable processor and memory resources.

All the articles in the book are concerned essentially with person-based linkage and with records which have a uniform structure. In a typical example using Italian fiscal records (Herlihy 1973) record linkage is presented as a very simple process, based on a comparison of entries in separate nominal lists. In fact, as for Blayo the computer was used only as an aid to manual linkage: in this case it was used merely to sort the nominal lists into alphabetical order.

Most of the articles examine the linkage of parish register records and the associated process of family reconstitution. This process has been defined (Wrigley 1973, 155) as the 'technique by which records concerning the vital events (birth, marriage and death) of all members of a family are linked to each other.' Fleury and Henry (2), working manually with French parish registers, were the first to attempt the formalisation of 'rules' defining the conditions under which such records should be linked. More recently, attempts have been made to incorporate such rules into computer programs designed to automate the process of family reconstitution (Wrigley and Schofield 1973; Skolnick 1973).

In so far as family reconstitution has traditionally been concerned solely with the linkage of birth, marriage and death events it represents a restricted form of record linkage. Nevertheless, it does involve the linkage of several types of records, and records which can typically contain the names of at least two family members. As such, it is an inherently more complicated operation than the purely person-based linkage which operates on entries from simple nominal lists. Thus, for example, the record linkage work pioneered by Wrigley and Schofield in the early 1970's has only recently succeeded in producing a working set of record linkage programs (Cambridge Group 1988).

None of the articles in 'Identifying people in the past' gives significant consideration to the linkage of more irregularly structured data sources, such as census household records. There is, therefore, no exploration of the developed, family-based concept of record linkage which is the focus of the present thesis. The use of Italian census materials is discussed (Skolnick 1973), but only as a means of checking the accuracy of parish register linkage.

Other, more recently reported linkage work has involved the use of civil registration and census data (Pearce 1973), poll books (Speck et al 1975; Mitchell and Cornford 1977) and poll books and subscription lists (Morris 1976). All of these were elementary person-based linkage studies, involving the merging of entries from separate nominal lists.

A much more ambitious linkage project, funded by the SSRC over a period of 10 years, was carried out by Alan Macfarlane at Cambridge University (Macfarlane et al 1983). The prime data used consisted of a wide range of source materials for the parish of Earls Colne in Essex from the period 1380-1750 (Macfarlane 1980A; 1980B; 1981). The data included parish register entries, wills, ecclesiastical court records and manorial records. Much of this data was irregularly structured, and this presented significant problems for its organisation and processing. In view of this, the linkage strategies which were adopted were directed towards the production of individual life-histories, rather than a more fully developed network of interconnecting families.

In the United States of America there has been significant linkage work carried out, notably by the Philadelphia Social History Project (Hershberg et al 1976A). The work of this project has centred around the record linkage of individuals across successive decennial population censuses for the city of Philadelphia. Again, the objective has been restricted to the production of life-histories rather than the more elaborate familial and residential structures of which the data is potentially capable. Linkage work along similar lines has been carried out over a number of years by the Canadian Social History Project (formerly the Hamilton Project), (Winchester 1970; Katz et al 1982). Other work on record linkage has been carried out by the Mormon Historical Demography Project (Bean et al 1977): this project has made use of American nineteenth century church records.

In parallel with the above projects, work has been progressing on the record linkage of present-day, rather than historical, records. The record linkage of health records is the most notable example (3), dating back to the Oxford Record Linkage Study, which was started in 1961 with a grant from the Nuffield Foundation (Wilson 1971). A more recent project of this kind involved the development of the Master Patient Index System for the Tayside Region (Graham, 1974). Such systems are essentially person-based, although in this case the absence of more elaborate family structuring may have less to do with the presence of insurmountable technical problems than with the confidentiality issues which surround the use of sensitive personal health data.

By contrast, in Canada an early medical project was able to establish the family-based linkage of present-day vital and health records from British Columbia (Newcombe 1966). The linkage operation was conceptually quite simple, involving a merging operation in which records from separate magnetic tape files were used to create a family master file. However, analysis of this file was able to furnish a measure of the family association of a wide range of diseases. This approach is still largely unexplored in Britain (Baldwin et al 1987, 336).

Increasingly, record linkage principles are being applied to present-day, non-demographic data, such as government and local authority records. LAMIS (Local Authority Management Information System) is a system, developed jointly in 1974 by Leeds City Council

and International Computers Limited, for organising all the information relating to housing and other property in a local authority area (International Computers Limited 1975). The system may be regarded as a 'linkage system' in the sense that it takes each 'spatial unit' (e.g. a house) and provides linkages to all items of information concerning that unit, wherever such information may be held, e.g. in rates files, plans registers, etc. As such, the system is functionally equivalent to the simple person-based linkage systems already discussed.

1.3 THE USE OF NOMINAL RECORD LINKAGE

Historical nominal records constitute a rich source of information about the demographic and social conditions of the past. An examination of the census enumerator's returns for a village in 1851, for example, can provide a wealth of information about the prevailing living conditions. Among the more basic items of information which can be derived from such a source are the total population size, sex ratio, age distribution, size of families, size of households, as well as the occupational and migratory characteristics of the population. Additional information, however, becomes accessible when one examines simultaneously the data about two or more co-residing members of each household. For example, one is able to study family and household structure (Anderson 1971, 81-6), and the birthplaces of the individual family members can provide an

indication of the migratory history of each family (op cit, 36-41).

Despite the richness of such a data source as the 1851 population census, it nevertheless presents for each individual and family only a single 'snapshot' at a particular point in time. What happened to each individual and family before and after the census was taken are clearly not revealed. Thus, although such a source can provide a myriad of 'snapshots' of people and families at all stages in their development, one is unable to examine the life-history of any one person or family.

It is the express function of nominal record linkage to overcome this limitation in the methods of using historical source materials. The aim is to take the individual 'snapshots' derived from separate historical sources, and bring them together to form a composite data source which will possess greater richness than the unlinked sources. (This is a manifestation of the Gestalt Theory, i.e. that the whole can be greater than the sum of the parts.)

Let us examine the informational implications of linking data, first within the context of person-based linkage. As already indicated, the main outcome from such linkage is the production of a personal dossier or life-history for each individual. Such a dossier can reveal information about the individual which it is impossible to derive from the isolated sources. For example, one can study the phenomenon of intra-generational occupational mobility, i.e. the degree to which a person's occupation changed and advanced during

their life-time. Or again with regard to occupation, one can study the relationship between occupation and age at death (4).

By subjecting the data to the more developed, family-based linkage one can produce a composite data source possessing even greater richness and which is able to yield information about a much broader range of demographic phenomena. The most important feature of the additional phenomena is that they are not restricted to individuals in isolation; rather, they can encompass the characteristics of people in relationships. So, for example, in the analysis of occupations it becomes possible to advance from observing intra-generational occupational mobility to observing inter-generational occupational mobility, i.e. one can examine, for example, the relationship between the occupation of an individual and that of his father. Or, again, as cited in the last section, if one is able to link health data for complete families then one should be able to examine the hereditary characteristics of a whole range of conditions, such as diabetes and heart disease.

It should be clear that record linkage, and supremely family-based linkage, provides a method of making optimal use of the source materials which are at our disposal. This is of particular importance in a historical context since there is no alternative, systematic way of examining the life and family histories of other than a very small fraction of the members of past populations.

1.4 THE OBJECTIVES AND SCOPE OF THE PRESENT RESEARCH

The primary objective of the present research has been to explore the use of novel computing strategies to achieve the family-based record linkage of historical demographic data. The emphasis has been predominantly on the methodology of record linkage, rather than on its application to a particular historical context and the answering of pertinent socio-demographic questions.

In order to ensure that this emphasis could be sustained it was decided to restrict carefully the amount of test data used, and this was achieved by operating within the constraints imposed by a quite limited historical and geographical context. This was as follows:

1. the period: 1812-71.

2. the location: two neighbouring small rural English parishes, Elwick Hall and Hart in County Durham (5).

3. types of data: parish register and household census (1851, '61 and '71).

4. size of study population: < 10,000.

The choice of the geographical study area was based on my own personal familiarity with the area and with the records, rather than on purely academic grounds. Its small size guaranteed that the data

preparation requirements would be modest.

The main reason for selecting a late study period, viz. the nineteenth century, was that this period provided data which was uniquely suited to the task of exploring the problems of family-based linkage. In particular, nineteenth century census data is a rich source, with considerable implicit complexity, and it was felt that the problem of achieving record linkage for families across three successive censuses (1851-71), as well as to the associated parish registers, presented significant and novel intellectual challenges.

It was decided that the most promising strategy for confronting the central issues of family-based record linkage would be to develop an operational record linkage system. It was considered that this development would provide a framework within which the associated problems could be addressed and corresponding solutions sought.

However, it was recognised that a potentially worthwhile secondary objective would be to design the linkage system in such a way that it might be readily used by other researchers to investigate study populations with characteristics similar to those identified above. The acceptance of this objective had a beneficial influence on the development in several ways. In the first place, it was seen as important to build into the system design as much flexibility and generality as possible. In practice this meant that the overall system had to be designed in a way which would facilitate its enhancement to deal with new, unforeseen requirements, for example to

permit additional types of records to be incorporated with minimal difficulty. Such considerations have prevented the research from becoming too narrowly focused on the specific problem under investigation. Other important features of the development included an emphasis on user-friendliness and systems integrity and portability.

A final objective of the present research concerns the form in which the material in this thesis is presented. In view of the interdisciplinary nature of the research topic it is obviously important that the material should be accessible to those in the associated disciplines. An attempt has been made, therefore, to present the thesis in such a way that both the socio-demographic historian and the computer systems designer will find it comprehensible and informative. This is clearly not a simple task, but it is hoped that the form of presentation will largely satisfy the disparate demands which may be placed on it.

In this outline of the objectives of the research it should be emphasized that during the formative stages of the project a number of careful decisions had to be made to ensure that the objectives would be achievable within the constraints of a PhD programme. Computer projects of this kind, particularly those which deal with 'awkward' historical data, are notorious for demanding considerable human resources, sometimes as much as 30-50 man-years. Thus, for example, Alan MacFarlane's record linkage project for Earls Colne, which ran approximately from 1971 to 1983, required significant human resources,

probably in the range 20-30 man-years (Macfarlane et al 1983). A considerable quantity of data was used (i.e. over three million words of text), and, in general, the records which were used were structurally more complex than those used in the present project. However, both projects were confronted with a number of common problems, in particular concerning the use of novel computing facilities, the provision of sophisticated linkage mechanisms and arrangements for structuring the linked data in a database.

In the present project, therefore, priority was given to the central issues of record linkage, and a number of interesting but peripheral areas of the problem were deliberately ignored. For example, it is generally acknowledged that a desirable feature of any record linkage system would be a facility which would enable the user to interact with and to exercise control over the linkage process (6). However, the design implications of such interaction are considerable, both for the provision of the user interface and, even more so, for the design of the underlying mechanisms which would support it. But even in purely practical terms the Edinburgh University computing system which I used (viz. ICL 2900 VME/B), was a batch system with input submitted on punched cards, and so in fact user interaction was never a real option. It is of note that the Macfarlane project was subject to a similar constraint (Macfarlane et al 1983, 21).

1.5 PROBLEM-SOLVING BY COMPUTER

Since this research is concerned with the application of concepts and techniques from one discipline (viz. computing) to the solution of a problem in a quite separate discipline (viz. historical demography), it is important at the outset to clarify the nature of the resulting interdisciplinary confrontation.

One of the perennial problems of such interdisciplinary endeavour is the task of finding a suitable marriage between two quite separate perceptual frameworks and sets of assumptions. Although even this statement of the problem is a simplification, since it suggests that a single, consistent view exists on each side of the intellectual divide. But as Becher has observed (Becher 1981, 115-6):

> Nearly all of those interviewed were at pains to emphasise that their disciplines, even if unified, were far from being homogeneous entities. Some went out of their way to draw attention to the complexity and variety of the approaches adopted, and the lack of any simple rubric to describe the activity in which they were engaged... Each of the areas of enquiry covered in the interviews has its rival factions and its competing ideologies; none represents a uniform, undivided set of beliefs.

Turning specifically to the application of computing to the solution of complex problems, a fundamental difficulty is that the people who are fully conversant with the nature of their problem are often not sufficiently aware of the range of computing facilities which are available, and so they are unable to take the appropriate steps to achieve a solution. Meanwhile for the computer systems analyst or designer the position can be equally difficult. Since he will not normally have a comprehensive grasp of the nature of the

problem he will find it difficult to establish which computing facilities and strategies will most likely lead to a viable solution (7). Inevitably the resolution of this difficulty requires some form of 'bridge-building' between the two disciplines, with ideas and perceptions being 'bounced' from one to the other in each direction.

Another feature of computer problem-solving which leads on from this is the nature of the process itself. How should one go about solving a problem using a computer? Superficially, one would imagine that the process would involve two distinct stages: problem analysis and solution implementation. In the first stage the objective would be to find out as much as possible about the nature of the problem. And in the second stage one would examine a whole range of computing 'mechanisms' (i.e. techniques, strategies, facilities, etc.) in an attempt to find those which, when employed together, would provide an appropriate solution to the problem. While the solution of elementary problems may roughly follow such a two-stage process, it seems that a quite different strategy needs to be invoked when confronting complex problems. In this case it is more fruitful to allow problem analysis and solution implementation to proceed together, with the emphasis shifting periodically from one to the other. In effect, this constitutes a restatement of the point made earlier of the necessity for 'bouncing' the problem and perceptions about it between the disciplines.

The logic of this more iterative approach is that a complex problem and its solution are inextricably bound, and that through the process of examining alternative solutions one can increase one's understanding of the nature of the problem. In its turn this increase in understanding can prompt one to examine a different class of solutions, and so the process continues and progresses.

An interesting example of this process of problem refinement concerns the use of database management systems (DBMS). Anyone who is searching for a computer solution to a problem may at some stage consider the use of a DBMS. What often happens when he does this is that the actual process of investigation forces him to re-examine his problem and restate it in terms which are meaningful within the conceptual framework imposed by the DBMS. Typically, for example, he may scrutinise his data more carefully than he has done so before, consider how it subdivides into individual record types, and he may possibly attempt to express the relationships between the record types by drawing a 'Bachman diagram', as described in Section 4.3.2. Interestingly, it may not be especially significant which DBMS he chooses to investigate, or, indeed, whether or not he subsequently decides to use it. The crucial importance lies in the stimulus which he is given to observe his problem in a new way.

The overall strategy adopted throughout the present research has been based on this concept of the inter-relationship between problem and solution. An understanding of the nature of the problem under investigation has therefore been established and developed largely

through the attempt to confront the problem as perceived with the computing mechanisms which are deemed to be appropriate. A central element of the research has been the objective of designing and implementing an operational computing system. This development has provided a framework within which the full complexity of the problem could be intimately experienced and within which the use of various computing facilities and mechanisms could be explored.

In part, this thesis contains a record of the complete problem-solving activity which has been carried out. As such, it provides a comprehensive examination of the record linkage problem, analysing both conceptual and strategic factors and also the more specific and pragmatic problems of implementation. Correspondingly, it provides a similar, comprehensive examination of the way in which these problems were addressed. Thus, there is an analysis of the overall design and implementation strategies which were adopted and an examination of the steps which were taken to resolve the range of issues presented.

It is important, however, that the research should also be viewed in a wider context. The project embodied in this thesis has been concerned with the challenge of investigating the applicability of new computing technology to a complex problem, and one for which earlier technology has proved inadequate. It is considered, therefore, that the overall approach adopted and lessons learnt will have relevance to other researchers who, in a similar way, are attempting to apply new advances in computing to complex problems in quite disparate

application areas.

1.6 THE SIGNIFICANCE OF CURRENT DATABASE TECHNOLOGY

In Section 1.2 it was observed that earlier work in nominal record linkage by computer has been restricted essentially to person-based linkage and to the linkage of records which have a uniform structure. This is no accident, but rather it reflects the fact that the predominant computing technology at the time was inappropriate for a more advanced approach to record linkage. Many of the early initiatives (e.g. Herlihy 1973 and Morris 1976) employed a technique commonly referred to as 'sort and merge' in order to achieve record linkage. This would typically be used to link simply structured records (e.g. poll book entries) held in sequential files, perhaps physically held on magnetic tape. During the first stage of processing, the records in each file would be sorted into alphabetical order by surname and forename. During the second stage, records from the separate files would be merged together to produce a composite sorted file. In the process of merging, therefore, records for the same individual, but originating from separate files, would take adjacent positions in the composite file, assuming that there were no name variations or ambiguities. The final stage of record linkage was often carried out manually, when it would require a human operator to scan a print-out of the sorted file to decide which groups of records should be considered to be linked.

It is quite clear that the primitive kinds of computing facilities which are described above are wholly inappropriate for carrying out family-based linkage and for the linkage of more irregularly structured data such as census household records. Each family consists of several members, and it is impossible to sort and merge the individuals, in the way described above, without ungrouping them and removing them from their family context. But, as observed earlier, such an operation would cause a loss of inter-relatedness and the informational richness which was inherent in the original source records (8).

It is fortunate that since these earlier developments computing technology has made significant advances, and there are now facilities available which are much better matched to the requirements of family-based record linkage. Flexible methods of structuring and handling records are currently provided by systems known as 'database management systems' (DBMS), and it is now no longer necessary to think merely in terms of simple sequences of records and operations such as sorting and merging.

In the business and commercial sphere a DBMS is typically used to integrate the handling of an organisation's machine-readable information. Whereas previously the organisation would have kept separate files for holding financial information, stock control information, etc., the DBMS effectively arranges to 'connect up' the isolated files into an integrated body of information, known as a 'database'. It should be clear that what is required for nominal

record linkage is precisely the same, viz. the connecting up of isolated files (e.g. baptisms, marriages, burials, census) into an integrated body of information. In a sense, therefore, a DBMS can be viewed as a 'record linkage machine' par excellence!

In the present research one of these systems, called IDMS (Integrated Database Management System), has played an important role in the implementation of the record linkage system. Although IDMS is not designed to make the decisions about which records to link, it does supervise the intricate storage and handling of records and the organisation of the interconnections between them.

It is conceivable that a family-based record linkage system could be implemented in a conventional programming language, without using a database management system such as IDMS. But in its place it would be necessary to produce some extremely complex programs for controlling the handling of the records. By using IDMS one is effectively provided with a very powerful programming language which is matched to the requirements of complex record-handling applications. The introduction of database management systems such as IDMS is therefore to be viewed as an event of crucial relevance to the development of nominal record linkage techniques. Its contribution to the present research is examined in later chapters.

1.7 THE STRUCTURE OF THE REMAINDER OF THE THESIS

Chapters 2 and 3 explore the problems of nominal record linkage. Chapter 2 begins by positing an 'ideal world' for record linkage, and considers the problems of implementing a linkage system in such an environment. Chapter 3 moves on to explore the nature of the additional problems in the 'real world' and the kinds of solutions which are appropriate.

The next two chapters examine implementation strategies appropriate to the solution of complex computer problems. Chapter 4 considers strategies of disaggregation and methods of simplifying the organisation of data and programs. It then provides an introduction to database management systems generally, IDMS and the concept of a population database. Chapter 5 confronts the more pragmatic issues which must be addressed and the choices which must be made when embarking on the implementation of an operational linkage system. It considers the selection of the implementation environment and the adoption of standard programming and database methods.

Chapter 6 examines the overall structure of the record linkage system which was implemented. It is essentially a preliminary systems analysis, with the emphasis being placed on <u>what</u> the system is required to do, rather than on <u>how</u> it might achieve it. The characteristics of the user interface are fully explored.

The following five chapters contain a detailed analysis of the design of the individual 'subsystems' which together form the total system. For each subsystem the functional requirements are identified, and the design of the corresponding strategies are discussed. Where appropriate the structural design of the database is also examined. The subsystems which are analysed in the five chapters are as follows:

7 Directory Prime

8 Source Translation
Source Loading

9 Record Linkage

10 Record Linkage (contd.)

11 Population Analysis

Chapter 12 examines the characteristics of the study population which was selected to furnish the data for testing the system and it analyses the results of the linkages which were carried out. The validity of the linkage strategies is assessed and a preliminary substantive analysis of the linked data is carried out. Finally, Chapter 13 provides an assessment of what has been achieved from the present research and makes a number of recommendations for future work in this area.

In view of the extended length of this thesis it is suggested that on an initial reading the following chapters should be omitted, or else skimmed over lightly: 5, 7-11. These chapters are for the most part concerned with detailed design and implementation issues, and it may therefore be appropriate to leave them aside for studying as a separate task. It should be added that the contents of these chapters, apart from the introductory sections, may prove to be of more relevance to the computer systems designer than to the socio-demographic historian. Thus, while the latter may be interested primarily in the broader, strategic questions, the former, particularly one who is setting out to develop similar strategies, will wish to obtain a much more intimate understanding of the ways in which the strategies may be established and made operational.

NOTES

1. A review of past work in nominal record linkage is provided in Section 1.2.

2. Fleury and Henry 1956, cited in Wrigley 1973, 64.

3. The concluding bibliography in Wrigley 1973 identifies a number of research initiatives in medical record linkage. Many of the associated papers were presented at a two-day international symposium on 'Record Linkage in Medicine', held in Oxford in 1967. The papers are collected together in Acheson 1968.

4. These analyses, and others, are explored in Chapter 12.

5. Within the time constraints of the project it was possible to carry out record linkage using the data for only one of the parishes, viz. Elwick Hall. This modification to the plan is discussed in Section 12.1.

6. Although some researchers have predicted, perhaps optimistically, that linkage systems, when perfected, will be fully automated (Katz and Tiller 1972, 145).

7. Such references to the designer, researcher, etc. as 'he' are of course to be interpreted throughout the thesis as signifying both male and female.

8. Modern data sources can be subject to a similar problem of inter-relatedness. For relevant comment on the use of the General Household Survey see Note 8 in Chapter 13.

CHAPTER 2 THE 'IDEAL WORLD' FOR RECORD LINKAGE

In this and the following chapter I shall provide a comprehensive analysis of the problems of nominal record linkage. In parallel with this analysis I shall also examine strategies which are capable of providing appropriate solutions. At this stage little attention will be given to the more computer-oriented aspects of the problem: an examination of such issues will be postponed until Chapter 4.

I shall develop my analysis of the problems of record linkage in a progressive fashion. In the present chapter I begin by positing an 'ideal world' for record linkage, and proceed to consider the problems of implementing a linkage system in such an environment. In the next chapter I shall move on to consider the nature of the additional problems which occur in the 'real world' and the kinds of solutions which are appropriate.

2.1 CHARACTERISTICS OF THE 'IDEAL WORLD'

Let me begin by constructing a picture of what one might call the 'ideal world' for record linkage. Given that one is aiming to link nominal records, what would be the characteristics of the ideal environment in which to do this? It is important to investigate this 'ideal world' for two reasons:

1. the 'ideal world' represents a base-line from which one can measure the problems of linkage in the 'real world'.

2. since the proposal is to use a computer to accomplish record linkage it will be valuable to see what are the central problems of implementation even in this 'ideal world'. Clearly, if the problems are substantial here then they could be overwhelming in the 'real world'.

The major characteristics of the 'ideal world' are as described in the following sections.

2.1.1 The Unique Identification of each Individual

As was observed in Chapter 1, when linking nominal records, one is faced with the problem of nominal ambiguity: the fact that a given name may be used by several people (homonymy), and that a given person may use different names on different occasions (synonymy). In such

circumstances record linkage is difficult, and there can be no guarantee that any link made will be 'correct'. The root of the problem clearly resides in the imprecision involved in identification by name.

If, alternatively, one imagines a situation in which each nominal record contains, not only the names of the individuals, but also some unique form of identification, such as National Health Service Numbers, then the situation is quite different. Identification now involves no ambiguity, and record linkage becomes simple and precise. This is the case even for the potentially complex family-based linkage, since one can readily create a unique family identifier by concatenating the two unique parent identifiers.

The Tayside Master Patient Index System, referred to in Chapter 1, uses such a unique form of identification (Angus et al 1978, 56-7). Each individual in the population is allocated a ten digit number, based on the person's date of birth and gender. Other digits within the number are used to distinguish individuals born on the same day and to provide self-checking should a number be entered incorrectly. The scheme is virtually identical to those which have been used for unique person identification in Sweden and Norway.

Such is the power of unique identification, and particularly when allied to the computer, that some people express fears that its widespread use in the future could facilitate the setting up of personal dossiers and an unwelcome invasion of the privacy of the

individual. Indeed, the Lindop 'Report of the Committee on Data Protection' devotes a whole chapter to the problem of the 'universal personal identifier' (HO 1978, 260-4).

Despite these apprehensions about the future, however, it is obvious that such identification schemes cannot readily be applied retrospectively to historical records, and so in this domain the problem of nominal ambiguity remains and must be confronted.

2.1.2 The Unique Identification of Places and Occupations

If in the 'ideal world' record linkage could be carried out by using unique person and family identifiers, then strictly it would be unnecessary to examine other information fields in the nominal records: linkages would be made (or not) purely on the basis of the values of the identifiers. However, if one were adopting a more generalised view of nominal record linkage then one might wish to link records on the basis of the placenames and occupations referred to in them. It would then be necessary to address oneself to the problems of uniquely identifying places and occupations.

Such problems, in fact, turn out to be even more severe than they are for personal names, and a detailed discussion of them will be postponed until the next chapter. For the present, one may simply observe that both places and occupations suffer from 'boundary problems' in a way that people do not. Thus, while a given individual

remains the same identifiable individual for the whole of his life, a particular village may, for example, at some point lose its individual identity (and possibly its name) and become part of a city. The same kind of problem applies to occupations. These do not possess discrete and static sets of characteristics, but, rather, they undergo change and development over time. And so a particular occupational title, say 'engineer', when used in the contexts of the 1851 and 1871 censuses, will not necessarily be associated with the same identifiable job on both occasions.

The question of unique identification is therefore rather complicated. For placenames a partial solution would be to include with them in the records their corresponding Ordnance Survey map references. Alternatively, and perhaps more conveniently, each identifiable location or area could be given its own identification code: such a scheme would be similar to the system of postal codes used by the Post Office. For occupations a moderately acceptable solution would be to include a code value derived from some occupational classification scheme: for example, the Standard Industrial Classification (CSO 1980). Unfortunately, while such identification techniques may be acceptable with present-day data, they cannot with confidence be applied retrospectively to historical records.

2.1.3 Perfect Record Formats

Record linkage can be considerably simplified if the records to be linked are deliberately designed so as to assist the linkage process. What then would perfect nominal records look like in the 'ideal world'? In the first place they would be rigorously standardised, i.e. there would be only one format for birth records, one for marriage records, one for census records, etc. A serious problem when dealing with historical records is that they come in a wide variety of shapes and sizes, and so require complex data-handling facilities.

Secondly, in the 'ideal world' there would be standard methods for representing particular types of information, e.g. ages and dates, and these would be used consistently across the different types of records. With historical records there is often inconsistent representation. For example, in post-1837 marriages the ages of the groom and bride are sometimes given as true ages, but also alternatively as '21' (meaning over 21), 'over 21', 'full age' and 'minor'.

Thirdly, each record would be designed to hold precisely the information that was required. Obviously such information would include codes for uniquely identifying individuals, places and occupations, as described above. But equally it would include other necessary information, relevant to the particular record. A serious disadvantage with historical records is that they usually contain too

little information. For example, baptism records normally do not contain date of birth information, and so they are an inferior substitute for genuine birth records. Equally, for example, it would be easier to link census data if household records were designed to carry information about family members (e.g. spouses) who were temporarily absent.

Finally, in the 'ideal world' the records would be designed so as to facilitate their input to the system. Pre-nineteenth century records are usually written in a 'free format' style, and, as a consequence, it is necessary to 'unpack' the information from each record and restructure it into a form suitable for input. Ideally, the original records should be clearly formatted so that they can be directly accepted by the system without the need for any restructuring. Moreover, given appropriate optical character recognition machinery, it should even be possible to input the original records directly into the computer.

2.1.4 Perfect Recording of Information

A final characteristic of the 'ideal world' for record linkage would be that the information contained in each record was guaranteed to be correct and complete, and that for every real-life 'event' there was a corresponding record. In short, this would mean that every event in the life of each individual was perfectly recorded.

In the real world data contains errors, caused either through misreporting or through incorrect entry; and, as a result, elaborate data validation procedures must be provided. But even where an error can be detected this does not mean that it can necessarily be corrected. It may be a simple matter, for example, to identify the presence of an error in a situation where a man's stated age is discovered to be less than that of one of his sons: but there may still be no way to correct it. Linkage systems must therefore be designed to be sufficiently flexible to cope with the residue of errors and inconsistencies which will remain in the data.

Finally, in the real world events can go unrecorded, for a variety of reasons, and so the results of record linkage will be incomplete. With Anglican parish register data, for example, there will probably be little or no reference to the vital events of nonconformists and Roman Catholics, and therefore, to some extent, the results of the linkage of such data will be incomplete and unrepresentative. These issues will be explored fully in the next chapter.

2.2 RESIDUAL PROBLEMS IN THE 'IDEAL WORLD'

Having analysed the major characteristics of the 'ideal world' for record linkage I shall now address attention to the problems of actually implementing a linkage system in this 'ideal world'. As it will be demonstrated, the problems are by no means trivial. In fact, they are typical of the computing problems which must be faced when developing any complex record-handling application. In the next chapter I shall proceed to consider in detail the characteristics of the 'real world' for record linkage and what additional problems must be confronted.

The residual problems in the 'ideal world' are as described in the following sections.

2.2.1 Data Input

A linkage system must contain facilities for reading in and 'unpacking' the nominal records in preparation for linkage. Since there can be a vast number of such records, e.g. birth, marriage, death, census, directory, tax, health, etc., corresponding procedures must be provided to handle the different record types and the various information fields within the records.

2.2.2 Data Processing

The form in which the data is most conveniently presented to the system will not necessarily be the most convenient for the subsequent processing. For example, the natural way of entering date information is to specify the day, month and year, e.g. '1 JULY 1989'. However, such a form is not particularly convenient for the system to handle when, for example, it has to compare two dates to see which one is the earlier. Therefore, it is common for systems needing to handle date information to convert each date into a more suitable 'internal' form. For example, the date '1 JULY 1989' could be converted into the number '32689', which is the number of days which elapsed between '31 DECEMBER 1899' (a reference date) and the date specified. Such 'computed dates' will clearly be more convenient when handling date comparisons, working out the elapsed time between two dates, and so on.

Other types of information, such as nominal, spatial, occupational and relationship information, can benefit from being converted after input into a more amenable form for subsequent processing. Consider relationship information, as specified in a census household record. This can be of two kinds: kin (e.g. father) and non-kin (e.g. lodger), and it can be useful for the system to derive the value of this attribute for each member of a household once-and-for-all, immediately after the record has been input. Such attribute values may then be stored with the record, and used when necessary.

The actual process of record linkage requires that the nominal records should be merged in some way, and that appropriate associations or linkages should be created. In the 'ideal world' there would, of course, be no problems of ambiguity, but it would still be necessary to provide procedures for accessing the records and organising the linkages. In practice this may prove to be a complex and formidable problem, but given appropriate database facilities the task can be considerably simplified.

2.2.3 Data Output

Facilities must obviously be provided to enable the linked information to be interrogated and analysed. Typically, one might wish to compare the attributes of two particular subgroups within the population, e.g. farmers and agricultural labourers. The system should therefore be able to respond to such specific requests for information, and furnish precisely the information required. This might, for example, be in the form of a table of results or else a file of data suitably structured for input to a statistical analysis package. Once again, database facilities can considerably simplify this task, and many database systems do, in fact, provide their own query facilities.

2.2.4 General Flexibility

In the 'ideal world' for record linkage, as portrayed above, no assumption was made that the 'systems' environment would remain forever the same. Therefore, in the 'ideal world' one would still need to be concerned with questions of flexibility, and ask how easily the linkage system could be adapted to cope with particular modifications in its environment. For example, would it be a simple task to move the system to a different hardware configuration? Or, again, could the system be readily modified to cope with new kinds of nominal records?

It would clearly be preferable to produce a system which has such flexibility, rather than one which is 'locked in' to a specific environment. But, conversely, one must bear in mind that general-purpose solutions are usually more difficult to implement than special-purpose ones.

The task of transferring a software system from one type of hardware environment to another can be difficult. However, if the system has been coded in a standard programming language, e.g. COBOL or FORTRAN, which is available in the two environments, then the task may be somewhat easier. Similarly, where database facilities are to be used, then it is preferable to employ a standard system, e.g. IDMS, which is widely available on different types of computers, rather than a system which is designed specifically for a particular machine.

Where there is a requirement to provide a linkage system which can be modified to cope with new types of nominal records, then it is important to adopt a programming strategy which will readily permit such flexibility. Good overall design and a clear subdivision of the system functions into separate subsystems and modules is vital. This subject will be examined in detail in Chapter 4.

CHAPTER 3 THE 'REAL WORLD' FOR RECORD LINKAGE

In the last chapter I investigated the problems of implementing a record linkage system in what I chose to call an 'ideal world'. In the present chapter I shall extend the scope of my analysis to encompass the additional problems which occur in the 'real world' and the kinds of solutions which are needed.

It is important to recognise at the outset that there is not just one 'real world' for record linkage. Broadly, one can subdivide linkage systems into two types:

1. those which deal mainly with present-day records

2. those which deal with historical records.

The first category includes systems which link modern health records. These were referred to in Chapter 1. In general, the task of developing such systems is considerably easier than for systems which handle historical records. In the first place, the problem of nominal ambiguity can be simply resolved by specifying that each linkable item (e.g. a person) is to be given a unique identifier, which must be included in each record. Secondly, the nominal records can be expressly designed so as to assist and simplify the linkage process. If, for example, a date of birth field would be of value

then it can readily be included in each record: when dealing with historical records one must be content with whatever information has been provided. Finally, with 'modern' linkage systems it is possible to reduce significantly the problem of dealing with error and exception conditions by employing conversational correction facilities. For example, if a particular name or identifier is found not to be meaningful to the system it can be rejected with a request to the operator to enter a more appropriate alternative. The operator may then, either see what is wrong and correct it himself, or else obtain the correct information from other people, and possibly from the person(s) referred to in the relevant nominal record (Winchester 1973, 134-5). When dealing with historical records it is normally not possible to correct the data in this way, and so the system must be provided with strategies which will enable it to cope satisfactorily with the information provided.

To summarise, with 'modern' linkage systems, where one has considerable control over the content and format of the records, it is possible to arrange matters so that the system will effectively operate in an 'ideal world', comparable to the one described in the last chapter. With 'historical' systems the situation is quite different. Here one has no control over the content and format of the records: the data is the 'given', and one must design the system accordingly.

But even for historical data there is not just one 'real world' and one set of problems. For any operational linkage system there

will exist a particular 'record universe', this universe consisting of the record types which the system has been specifically designed to link. It is the case therefore that when one chooses a particular 'record universe' one simultaneously selects for oneself a particular 'real world' and an associated set of problems.

Let us compare the linkage problems faced in the following two quite different historical contexts:

1. the seventeenth century, with the nominal data being provided from parish registers.

2. the nineteenth century, with the nominal data being provided from census household records.

In the first context the records usually contain relatively little information, and the standard of reporting is often variable. As a result, the matching of records and the carrying out of person-based linkage can be difficult, with nominal ambiguity posing a particular problem (1). By contrast, in the second context there is a plentiful supply of information in each record, and nominal ambiguity is a much reduced problem. Instead, the major difficulty is the organisation of family-based linkage across the censuses and the handling of the structurally complex data.

The net conclusion to be drawn from this preliminary analysis is that historical record linkage presents a broad range of 'real

worlds', each with its own distinctive characteristics and problems. From a systems design perspective therefore the appropriate response is to analyse the characteristic properties and inherent problems in each 'real world' and design a system accordingly. The alternative approach, of seeking a generalised solution to handle all 'real world' linkage contexts simultaneously, seems at present to be unrealistic, given the extent and seriousness of the associated problems.

In the following sections I shall provide a comprehensive analysis of the problem areas encountered when carrying out nominal record linkage with historical data, together with proposed solutions. The first two major sections contain an examination of the problems of identification and ambiguity, while the final two sections address the broader issues concerning the form and content of the original source records. For the present I shall concentrate mainly on the issues which relate to the problems of person-based linkage. In later chapters (and specifically in chapters 9 and 10) I shall widen the context of my enquiry to encompass the more complex problems of family-based linkage.

3.1 NOMINAL AMBIGUITY

3.1.1 Problem Description

Nominal ambiguity is the problem which has traditionally been associated with nominal record linkage. Quite simply, the problem is that the normal method by which people are identified in nominal records, viz. by their names, is inadequate and open to ambiguity. In the last chapter reference was made both to the problem of homonymy, the fact that a given name may be used by several people, and also to the problem of synonymy, the fact that a given person may use different names on different occasions. The change of a woman's name at marriage is an additional and somewhat anomalous problem, which can be regarded as a particular variant of the synonymy problem.

One can summarise the problem of nominal ambiguity in the following way. Ideally, the relationship between people and names would be 1:1, i.e. for each person there would be only one name, and for each name there would be only one person. In reality, the relationship is n:n, i.e. for each person there can be many names, and for each name there can be many persons.

3.1.2 Solution

Let us initially consider the problem of homonymy. Given that one may encounter several records which contain the same name, say 'John Smith', how can one determine whether they all refer to the same individual or to several people called 'John Smith'? There is no simple answer to this question and no logical sequence of tests which can be applied. The problem, like the data, is complex and idiosyncratic, and it is necessary initially to seek some simplifying concepts. The solution to the problem appears to require two basic stages (Winchester 1970, 113). During the first stage one would subject the records to a simple sequence of tests, both logical and demographic, which would serve to filter out impossible linkages. Having done this, one would then examine each remaining potential link, and attempt to evaluate its 'correctness' in the light of all the available and relevant information.

Consider the first stage and the logical tests which might be carried out on the records. In the first place, multiple occurrences of a particular name in a single record can serve to indicate multiple occurrences of individuals with that name. For example, if one locates a baptism record for a 'John Smith, son of John Smith', then one clearly has evidence for the existence of two distinct people with the same name. Similarly, a census household record which contains two references to 'John Smith' again constitutes conclusive evidence for the existence of two people with the same name, and such occurrences should obviously not be linked. All this is

straightforward.

Secondly, for many types of records there is a strict 1:1 relationship between people and occurrences of that record type. Consider burial records. Since each person has, at most, only one burial event, if one locates two 'John Smith' burial records then they must obviously refer to two separate individuals. The same 1:1 relationship holds for census records, since on a given census night each individual can be resident in only one place. Again, multiple occurrences of 'John Smith' would indicate that there were multiple occurrences of individuals with that name.

Thirdly, there exist constraints between records of different kinds. The simplest concerns the chronological sequencing of birth and death records, and it dictates merely that a person's date of birth must not be later than his date of death. Similarly, every other record relating to an event in the life of an individual must obviously occur at some point between the person's date of birth and date of death.

Finally, there are some additional constraints associated with particular types of records. Consider the marital status information contained in a marriage record. If the status of the bridegroom is given as 'bachelor' then this indicates that he has not been married before. If, instead, the status is given as 'widower' then this indicates both that he has been married before and also that his previous wife is dead. Therefore, the values of the marital status,

for both groom and bride, will constrain the ways in which each marriage record may be linked to other records.

In addition to applying such logical tests during the first stage one can also apply demographic tests. For example, if one has a 1734 birth record and an 1854 death record for 'John Smith', then obviously logically they can be connected together. However, one might still decide that the records could not reasonably belong to the same person, since if they did it would imply an age at death of 120. Most other demographic tests are also concerned with ages, and specifically with the age-ranges within which particular events, such as child-bearing (for women) and marriage, are expected to occur (2).

Having applied the above logical and demographic tests, and having thereby filtered out both the impossible and the improbable linkages, one should then be left with a relatively small number of potential links. It will now be necessary, during the second stage of the tests, to decide which of these links, if any, should be deemed to be correct. The strategy for making this decision will need to be appropriate to the particular historical context which is being used and the nature of the records being linked. In general, however, the method will need to involve some kind of matching of the various information fields in the records. For example, when attempting to match people recorded in both the 1851 and 1861 censuses it will be worthwhile comparing the place of birth information for any potential link.

Especially valuable for matching purposes is age and date of birth information. Given a person's age in years, and assuming that it has been reported and recorded correctly, one is able to calculate their date of birth to within an accuracy of six months (3). Such information can usually help to resolve most matching problems. However, where one is supplied with the date of birth itself then one has a most powerful piece of information for matching purposes. In fact, such is the discriminating power of this information that if one takes a person's name and concatenates it with their date of birth then this will normally produce a unique combination (akin to a universal personal identifier), even in a population as large as 50 millions (4).

Other information which is valuable for matching purposes, and which is often present in nominal records, is information about a person's relatives. A typical example is the information about parents, which is given in baptism and birth records. And in census household records there can be information about many more relatives: for example, details of a person's brothers, sisters, grandparents, etc. Indeed, such is the richness of this particular source that it is no longer appropriate to constrain its use merely to person-based linkage operations. It is therefore necessary to widen such concepts of record linkage to encompass linkage at the higher and more complex, family-based level, and I shall explore the implications of this in later chapters.

There are two types of information which one should be careful to ignore when matching records: those relating to a person's residence and occupation (Katz and Tiller 1972, 146). And the reason for ignoring them is that they are not invariant over time, i.e. people move around and people change their occupations. If one did decide to use such information as part of the matching strategy, then in one's results one would risk underestimating the true levels of geographical and occupational mobility in the population. By using the information one would be implicitly saying that, when in doubt, one should assume that people did not move around and did not change their jobs. While such a conclusion might possibly arise from an analysis of the linked data, it should clearly not be built a priori into the linkage strategy.

To summarise then the method for dealing with the problem of homonymy. First, it is necessary to apply a series of logical and demographic tests on all potential linkages so that impossible and improbable links can be identified and subsequently ignored. For each remaining link one must then use some form of matching strategy to determine the correctness of the link. This strategy will make use of all relevant information in the records: time-variant information, such as residence and occupation, should, however, be ignored.

I shall now turn to the second major problem, i.e. that of synonymy. It should be noted at the outset that I shall be ascribing a somewhat wider meaning to the term 'synonymy' than is contained in its precise dictionary definition. Strictly, two words are regarded

as synonyms if they share a common meaning and usage. Thus, the Christian names 'John' and 'Jack' are true synonyms, and they are, in fact, often used interchangeably when referring to the same person. Surnames also fall into synonym groupings, and the surnames 'Smith' and 'Smythe' would normally be regarded as synonyms. However, in practice, there is much less likelihood of surname variants being used interchangeably, and, when it does happen, it probably occurs more by accident than by intention. For example, a registrar who happens to be unfamiliar with a particular surname may consistently use one of its synonyms, or even perhaps some name which vaguely resembles it.

A record linkage system must obviously be designed to cope with the kinds of name variations described above, and, for example, it should be able to recognise that 'John Smith' and 'Jack Smythe' are potential variants of the same name. However, there are two additional name-recognition problems which are, in a sense, problems of synonymy. Reference has already been made to the change of a woman's name at marriage. In order to cope with such a change the linkage system will need to be capable of registering the fact that two quite different female names, such as 'Ann Hutchinson' and 'Ann Brown', may in a particular context be synonymous, i.e. refer to the same person. An additional name-recognition problem occurs with nicknames and other aliases. Unfortunately, for all practical purposes this problem is insoluble. Aliases are usually given to people in a quite private way, and there is often little rational or meaningful connection between a person's real name and his alias name. Unless, therefore, one locates a record in which both the name and the

alias appear together, it is unlikely that the connection will ever be made. The alias problem can also occur with people who have two or more Christian names. For example, a person whose true name is 'John William Taylor' could be referred to in different records as 'John Taylor' and as 'William Taylor', and again the linkage of such aliases could prove difficult, if not impossible.

A commonly adopted solution to the problem of name variation is the use of a special coding algorithm, the most widely known, perhaps, being Russell-Soundex. The function of such an algorithm is to scan the letters of each name and convert them to a code value, the intention being that genuine synonyms will convert to the same value. For example, the names 'SMITH', 'SMYTH' and 'SMYTHE' do, in fact, all convert to the same Russell-Soundex code, viz. S530, and so they can accordingly be assumed to be synonyms (5).

Unfortunately, such algorithms cannot be guaranteed to produce the desired result in every situation, and it has been observed (Blayo 1973, 57) that the basic Russell-Soundex coding scheme is unsuitable for handling French surnames. Sometimes genuine synonyms produce different codes, e.g. the English surnames 'THOMPSON' and 'THOMSON' produce codes T512 and T525, respectively. Conversely, quite different names can convert to the same code, e.g. 'MOSS' and 'MACKIE' both give M200. Additionally, since Soundex methods operate on the assumption that names which sound similar are to be regarded as synonyms, they will obviously fail when confronted with any synonym group whose members do not sound the same. The two Christian names

'John' and 'Jack' are examples of such names, as also are the names 'Frances' and 'Fanny', and a Soundex algorithm could obviously not be expected to cope successfully with these (6).

In view of the limitations and unreliability of Soundex coding methods it would clearly be unwise to incorporate them into a fully automated system. The problem of name variation is not wholly susceptible to rational analysis, and no algorithm will ever be able to deduce when two arbitrary character strings are to be regarded as synonyms. The need for some form of human interaction is therefore inevitable. Indeed, the only kind of situation where Soundex methods appear to be completely successful is in real-time applications, such as airline booking systems. Here there is normally an operator monitoring and interacting with the system, and generally making sure that the operation of the Soundex algorithm does not lead to incorrect identification.

An alternative to the Soundex approach is to use name directories. In this case the system will no longer be required to make judgments about whether two particular names are synonymous. Rather, the user will be responsible for making such decisions, which he will then input to the system (7). For example, if the user decides that 'John', 'Jack' and 'Jonathan' are to be treated as synonyms then he might submit this information in the following way:

CN/M/JOHN/JACK/JONATHAN

The initial 'CN' tag would indicate that information about Christian names was being provided, and 'M' that the following synonyms were to

be regarded as male. The system would then store this information in its directory of Christian names, and use it as necessary.

An obvious advantage of the directory approach is that additional information, such as the gender, in the case of Christian names, can be readily maintained in the directories, and used for checking and other purposes. Soundex-type schemes make no allowance for such additional information. Further, when dealing with certain kinds of names, such as the names of relationships in census records, it becomes essential that a directory-type system should be employed. Thus, in the case of relationships it is not sufficient merely for the system to be able to recognise that 'DAU' and 'DAUR' are synonyms for 'DAUGHTER', it must also be able to attach meaning to these names so that it can make the appropriate family connections. For this purpose the use of some kind of table or directory is essential.

A scheme for organising name directories has been devised for the present project, and it will be described fully in Chapter 7. For the present, we may briefly observe how the code values which it employs are used to provide a synonym recognition capability. Consider the three Christian names 'John', 'Jack' and 'Jonathan', which, it is supposed, are to be treated as synonyms. After input to the directory system they are given code values by the system, such as 00031800, 00031801 and 00031802, respectively. Each code value is unique and can be used as an alternative for the actual name. However, these code values possess an additional important property which facilitates the detection of synonyms. The first six digits of any code value

represent a 'major' code, which will be the same for any group of synonyms. Thus, 'John', 'Jack' and 'Jonathan' all have the same major code, 000318, and so they are shown to be members of the same synonym group. The final two digits can be used to distinguish one synonym from others in the same group, whenever this is necessary.

Although such a directory approach puts an extra burden on the user, in practice this turns out to be not as irksome as it might appear. And, in return, of course, the user is given complete control over the handling of names. The amount of name information that needs to be submitted is small in comparison with the total information contained in the source records, and this is especially so for large populations. Thus, if there were 100 or even 1000 occurrences of the Christian name 'John' in the source records then it would still only be necessary to submit the name once to the directory system.

The remaining problem of synonymy is the change of a woman's name at marriage. In the example given earlier it was suggested that two different female names, such as 'Ann Hutchinson' and 'Ann Brown', might in a particular context be synonymous, i.e. refer to the same person. This problem is quite different from the other name variation problems, and it will clearly not be soluble by Soundex or directory approaches. There are, however, two distinct ways in which the problem can be tackled.

The most obvious way of deducing a woman's synonym(s) is to locate her marriage record(s). Each such record contains the bride's

pre-marriage name (which will normally be her maiden name) and also, if not explicitly then by inference, her married name. The marriage record therefore provides the link between the bride's two names, and so enables any pre-marriage and post-marriage information to be linked together.

The second method of deducing a woman's synonyms is a more generalised approach, in which synonym identification is inferred from the names of relatives referred to in a particular record. Thus, for example, if one has a census household in which there is a male head of household and a married daughter, then one may infer that the married daughter's maiden surname will be the same as the head's surname. A similar kind of inference can be made in the situation where a married woman is living in a household in which the head is either her brother or an unmarried sister. However, it should be noted that there is a certain risk involved in making such inferences. Relationship information is sometimes incorrectly entered, and more complex relationships, such as step-relationships, may not always be entered as such. Should a step-daughter, for example, be entered as 'daughter', then the inference made about her maiden-name will almost certainly be incorrect (8).

3.2 SPATIAL AND OCCUPATIONAL AMBIGUITY

3.2.1 Problem Description

The problem of nominal ambiguity is not restricted to people's names: in fact, it affects all named objects. Therefore, when handling the various non-name information in nominal records one must be prepared to deal with the homonymy and synonymy problems which will inevitably arise. However, when one looks more closely at particular types of information, and for the moment I shall be specifically concerned with spatial and occupational information, one finds that there are additional ambiguity problems, requiring special treatment.

Consider first the identification of places. And it should be immediately evident that the relationship between places and placenames is n:n, i.e. for each place there can be many names, and for each name there can be many places. Thus, for example, the city of Newcastle upon Tyne is sometimes just referred to as 'Newcastle' (synonymy); and at the same time there are three distinct places in England called 'Newcastle' (homonymy).

But consider a further problem. When a person is asked to give his birthplace, for example, during a census, it is possible for him to provide this information in a variety of forms. A person born in Elwick Village in County Durham might give his birthplace either as 'Elwick Village' or 'County Durham', or, again, as 'North Urn Farm' or

'Hart Parish'. While each of these answers might be correct, the places referred to are obviously not identical. The problem is one of precision, and a system which is required to match birthplace information which has been given with differing degrees of precision will need to be designed in a moderately sophisticated way. Taking the example given above, the system would need to be primed with the information perhaps that North Urn Farm was in Elwick Village, and that Elwick Village was in Hart Parish, and finally that Hart Parish was in County Durham. And when one considers that some villages cross parish boundaries and some villages and parishes cross county boundaries, the matching problem can become very complicated indeed.

Another similar problem associated with placenames, and one which was briefly referred to in the last chapter, is that of changing boundaries and changing identities. Over a period of time a particular locality, say a village, can be absorbed into a much larger unit, for example, a city. This can add considerably to the name-matching problem, particularly where places are renamed.

Turning now to occupational information one can see that the ambiguity problems which arise closely parallel those already described for spatial information. Thus, there are problems of synonymy, homonymy and precision. A particular occupation may be given two different names (synonymy): for example, 'farm servant' and 'hind' may be used interchangeably. Furthermore, a particular occupational title may refer to two distinct occupations (homonymy): for example, 'doctor' can refer to a qualified medical practitioner

or, alternatively, to the holder of a university higher degree. Finally, a person employed as a ploughman could be described as an 'agricultural labourer' or even merely as a 'labourer' (precision).

Occupational descriptions, like placenames, also suffer from boundary and identity problems. For example, what, if anything, distinguishes the three occupational titles of 'seaman', 'sailor' and 'mariner'? Also, if there is a difference between these, has this difference always been universally recognised and applied? Unfortunately, the answer is that occupational titles are not used in a rigorously defined way, and that over a period of time and in differing geographical locations the same title will not necessarily refer to the same identifiable job. It follows that any conclusions which are eventually reached about the occupational characteristics of a population should be accepted in a somewhat tentative and critical manner, and this has been borne out by some findings from the present research (9).

To sum up the problems of spatial and occupational ambiguity. As for personal names, placenames and occupational titles are prone to synonymy and homonymy problems. Additionally, they suffer from variable precision, and complex procedures are needed to enable precise and imprecise designations to be matched. Finally, and most intangibly of all, they suffer from boundary and identity problems. Thus, for example, given that one can somehow determine that a particular person worked as an agricultural labourer in a particular village for the whole of his life, one may still not know what he

actually did and where precisely he lived during this time.

3.2.2 Solution

The problem of synonymy is the simplest to deal with, and the setting up of appropriate directories, as for Christian names, can be readily implemented. Suppose, for example, that one wishes the directory system to treat 'Newcastle upon Tyne' and 'Newcastle' as synonyms, then one might input this information as follows:

PN/NEWCASTLE UPON TYNE/NEWCASTLE

In this case the initial 'PN' tag would indicate that information about placenames was being given. Correspondingly, if, for example, one decides that 'hind' and 'farm servant' are to be regarded as occupational synonyms then one would input this information as follows:

JN/HIND/FARM SERVANT

Here the initial 'JN' tag would indicate that information about job-names was being given (10).

The problem of homonymy is rather more complicated to deal with. Consider, again, the placename 'Newcastle'. As already mentioned, this name could refer to one of three distinct places in England: Newcastle upon Tyne, Newcastle under Lyme and Newcastle in Shropshire. The information that would be submitted to the placename directory about 'Newcastle' might therefore be extended as follows:

PN/NEWCASTLE UPON TYNE/NEWCASTLE/NEWCASTLE UNDER LYME

The placename directory would subsequently and quite correctly treat 'Newcastle' as a synonym both of 'Newcastle upon Tyne' and of 'Newcastle under Lyme'. Unfortunately, it would incorrectly treat 'Newcastle upon Tyne' and 'Newcastle under Lyme' also as synonyms, which they certainly are not. The net effect would be that when comparing birthplace information 'Newcastle upon Tyne' and 'Newcastle under Lyme' would be regarded as a good match.

This defect in the simple directory approach arises because of the problem of homonymy, the fact that several distinct places share the same or a similar name. Regrettably, there is no simple solution. In some records, such as post-1841 census records, birthplace information normally includes the county name, and so the use of this could enable certain placename ambiguities to be resolved. For example, 'Newcastle, Staffs.' and 'Newcastle, Salop' could be distinguished in this way, but, of course, there would still be no solution to the problem of two places with the same name being located in the same county. The implementation of directory facilities to handle the additional county information would also be complicated, and difficult to justify in view of the limited usefulness of such facilities.

The problem of homonymy when related to occupations is not quite so severe, chiefly because the vast majority of occupational titles are unambiguous and do not refer to more than one distinct occupational activity. But there are notable exceptions, such as 'crofter', which signifies quite different occupations in the north of

Scotland and in the textile areas of Lancashire (Anderson et al 1977, 120). In order to resolve such problems Anderson adopts the use of supplementary, context-specific directories in place of a single occupational directory. For our present purposes, however, it should be added that, since occupational information should not normally be used when matching records anyway there is no requirement at the linkage stage that name ambiguities should be resolved.

Returning again to placenames, we must now consider the problem of precision, and how it can be solved. In the example cited earlier it was suggested that an individual might on different occasions give his birthplace as 'North Urn Farm', 'Elwick Village', 'Hart Parish' and 'County Durham'. And the requirement was that the system should be able to recognise these as compatible alternatives.

In order to solve this problem the system needs to have at its disposal a certain amount of geographical information, enabling it to conclude, for example, that 'North Urn Farm' and 'Elwick Village' are not mutually exclusive places, while 'North Urn Farm' and 'Newcastle upon Tyne' are. There are several ways in which such geographical information could be organised. Typically most schemes operate by dividing the total geographical space into a series of zones and then associating each place with one or more of the zones. In order to match two places it is then necessary merely to compare the zone(s) associated with each place, and check whether they have at least one zone in common. If they do not, then one can conclude that the places are mutually exclusive; otherwise one can accept them as compatible

alternatives.

A highly sophisticated scheme would use a large number of zones, perhaps adopting each Ordnance Survey grid-square, 1 km. x 1 km., as a zone. For an urban area one might wish to employ an even smaller zone size. Thus, for example, the Philadelphia Social History Project uses a zoning scheme based on the 'grid block', a rectangular areal unit measuring 660 feet by 775 feet (Hershberg et al 1976B, 100). This kind of scheme is potentially very powerful, but it can also require considerable effort to organise. For the purposes of the present research a more basic alternative has been designed, which, although simple, is effective and is capable of demonstrating the zoning technique.

The present simple scheme uses only three zones, as follows:

1. the 'B' or base zone. This will typically be the immediate study area in which one is interested, and will consist, perhaps, of two or three contiguous parishes.

2. the 'N' or non-base zone. This consists of the whole area outside the base zone.

3. the 'P' or peripheral zone. This consists of the total area, i.e. 'B' + 'N'.

Each placename, together with its synonyms, is allocated to one of the

above zones by including the appropriate zone character as part of the input to the placename directory. For example, 'North Urn Farm' and 'Newcastle upon Tyne' would be allocated to the base and non-base zones respectively, as follows:

PN/B/NORTH URN FARM

PN/N/NEWCASTLE UPON TYNE/NEWCASTLE/NEWCASTLE UNDER LYME

Subsequently, when matching birthplaces, one can use the zone code as a discriminator: thus, if the codes for two places are not identical then one can conclude that the places are mutually exclusive (11).

There are several reasons for defining a 'P' (peripheral) zone. Firstly, there can be places which are geographically close to the boundary of the base zone, and which, as a result, can cause confused matching. For example, where there is a village immediately outside the base zone boundary it is likely that an individual born within the base zone, but close to the village, will occasionally have given the village as his birthplace, and at other times a more precise base zone location. To avoid mismatching such placenames one can define the village as a peripheral zone place: as such, it will be accepted as a suitable match for places within and outside the base zone. A similar situation occurs where homonyms exist on both sides of the base zone boundary. For example, if there were two places called 'North Urn Farm', one within the base zone and the other outside it, then one would be unable to associate the placename exclusively with either of the two zones. In this case it would be necessary to associate it with <u>both</u> zones, and so one would designate it as a peripheral place.

As has already been mentioned, occupations are not normally used when linking records, and therefore there is less need for sophisticated methods for handling occupational information. However, a synonym facility is valuable when access to the linked data is required. In this situation one may, for example, wish to obtain information about people with a specific occupation, and for this purpose all variants of the occupational title should be automatically grouped together.

In an earlier example it was suggested that a suitable method for specifying that 'hind' and 'farm servant' were to be designated as synonyms would be to input the information as follows:

JN/HIND/FARM SERVANT

Since occupational titles, like Christian names, are often exclusively male or female, one could usefully include such gender information, together with the synonyms. For example, if one wished to designate 'housemaid' as a female occupation then one could do this as follows:

JN/F/HOUSEMAID

Subsequently, the system would be able to check the mutual compatibility of this occupational title vis-a-vis the other personal attributes in a record. Should it, for example, encounter a housemaid called 'William', and assuming that 'William' had been designated as a male Christian name, then it would be able to issue a warning to the effect that a gender incompatibility had been detected.

3.3 INCONVENIENT RECORD FORMATS

3.3.1 Problem Description

Before record linkage can take place it is necessary to transfer information from a variety of source documents and set it up in the computer in a way that will facilitate the linkage process. Although this operation, sometimes referred to as 'data capture', may sound trivial, in reality careful thought must be given to it if subsequent data processing problems are to be avoided. Among the issues which must be addressed are: the type of input medium, record and field formats, validation, error-handling and the encoding of text fields into numeric codes (Ginter, Grogono and Bode 1977).

If one were to decide, for example, to input the information precisely as it occurs in the source documents then there would be little difficulty in actually getting the data into the computer. However, the subsequent programming task of interpreting and attaching meaning to the data would almost certainly be insurmountable. Computers are not good at processing such 'free formatted' text, other than in a rather trivial way, e.g. counting the number of words in each sentence, or finding the number of occurrences of particular words. If, therefore, one is to make the programming task of interpretation more manageable then one must take steps to present the data to the computer in a more convenient and 'pre-digested' form. To understand what this will mean in practice it is first necessary to

examine some typical source records and analyse the problems associated with them.

Consider the following sequence of baptism records as it appears in the Elwick Hall parish register (DCRO 1978 Vol 2, 15):

1796

> Hannah Daughter of Thomas and Frances Ainsley of Elwick Hall was baptized March 7th.
>
> John Son of John and Frances Shotton of High-Barn was born March 21st. and baptized March 23d.
>
> William and Mary twin Children of Thomas and Jane Wheatley were baptized April 5th.

In the first place it will be obvious that these records are presented in a free formatted, natural English style: as such they are difficult for a computer to handle. Note, in particular, that no two records have the same informational content. The first simply records the occurrence of a baptism; the second, although similar, provides additionally the date of birth of the child. Finally, the third records the baptisms of twin children. Writing computer programs capable of interpreting such varied structures is extremely complicated. And were one to consider handling the even more amorphous kinds of nominal records which are available, e.g. wills and marriage settlements, then the problems would be yet more acute, as has been observed by Macfarlane (12)..

A second problem with nominal record formats is that they change radically over time. Turning again to the Elwick Hall parish

register, let us compare the following two baptism entries with those displayed above (13):

> Thomas Wheatley, born April 29th., baptised April 30th. 1798, 2'd Son of Thomas Wheatley, Farmer, native of Garmondsway Moor, by his Wife Jane Fawell, Daughter of Anthony Fawell, native of Lanchester.
>
> Jane Ainsley, born Feb. 15th., baptised March 5th. 1799, 4 Daug'r of Thomas Ainsley, Hind of Elwick Hall, native of Sedgefield by his Wife Frances Forster Daug'r of John Forster native of Witton Gilbert.

The increase in information content of these records is dramatic. Not only is there now the date of birth and position in family of the child, but also the father's birthplace and occupation and the mother's maiden name, birthplace and the name of her father. Indeed, many of the baptism entries at this time include the names of all the child's grandparents. Now, while such additional information is potentially very valuable for making decisions about linkage, it also clearly introduces extra processing complexity. In particular, the system needs to handle two distinct styles of baptism entry and their correspondingly differing information contents. And after 1812 a further style of entry was introduced, thus compounding the problem. Similar changes have occurred with marriage, burial and census records, and so the problem of coping with such a wide range of record styles can be a considerable one.

A third problem with nominal records is that they often do not contain enough information, or, rather, they do not contain precisely the kinds of information which are required. Considering once again

baptism records, these are commonly employed by socio-demographic historians as surrogate birth records for periods when real birth records are not available. Unfortunately, such records do not normally contain the person's date of birth, and so some kind of estimate has to be made, based on the date of baptism provided and the likely interval which will have elapsed between birth and baptism. The uncertainty introduced by such estimates has a detrimental effect on the measurement of fertility and mortality (Schofield and Berry 1971; Razzell 1972; Wrigley 1975), and only serves to weaken the potential usefulness of record linkage work based on pseudo- rather than actual vital records.

Lack of information is also a problem with post-1812 burial records. Consider the following entry from the Elwick Hall register (DCRO 1978 Vol 2, 137):

1828

Name	Abode	When buried	Age
Isabella Walker	Elwick	April 4th.	75

Clearly, the age provided here is a very valuable indicator of when approximately the person was born. However, apart from this, there is little other information which will enable one to decide whether or not one should link this record with other 'Isabella Walker' records. In particular, since the record does not give Isabella Walker's marital status or the name of her husband there is uncertainty as to whether 'Isabella Walker' is a maiden or married name. Under these

circumstances there could be some considerable difficulty in making a correct link from this record to the corresponding baptism record.

A final problem with nominal record formats is that there is often an inconsistency in the way that a particular type of information is presented in different records. Consider the ways in which a person's age can be expressed. Reference has already been made in the last chapter to the variety of ways in which ages are given in marriage records, i.e. they are sometimes expressed as true ages, but also alternatively as '21' (meaning over 21), 'over 21', 'full age' and 'minor'. In burial records ages, where given, are usually expressed as true ages, in the form of an integral number of years, months, weeks, days or hours. Alternatively, they may sometimes be given in a more complex form, e.g. '1 year & 7 months'. For an old person the age might be expressed in an approximate manner, e.g. 'about 90', while for a young child the term 'infant' is often used. Finally, as has already been mentioned, adult ages in the 1841 census are expressed in a somewhat anomalous fashion, in that they are rounded down to the nearest multiple of five years. Thus, a person whose true age was 28 would appear in the census as aged 25. All of these idiosyncratic modes of representation must be accommodated, and this will inevitably lead to an increase in computer programming complexity.

A second example of inconsistent presentation of information is in the specification of geographical locations. For example, residence information in post-1754 marriage records normally contains

the name of the parish, possibly with an additional and more precise geographical identification. However, in baptism and burial records there is no standard method of presentation: locations are usually described briefly, e.g. 'Elwick', but are sometimes more detailed, e.g. 'Dovecot House in the Parish of Hart'. Turning to the birthplace information contained in post-1841 census records one finds that locations within Great Britain are normally specified by the county and the place within the county. But for the 1841 census such detailed information is not provided, and instead there is only an indication of whether the individual was or was not born in the county of his current residence.

A final example of inconsistent presentation of information is in the specification of occupations. Parish register entries normally contain brief designations, e.g. 'Farmer' and 'Joiner'. However, in the census quite detailed occupational descriptions are usually given, e.g. 'Farmer of 212 acres, employing 3 men and 2 boys'. The provision of facilities to process these more detailed descriptions can, again, introduce additional significant computer programming complexity.

To sum up one can say that the structure and content of historical nominal records is not geared to easy manipulation. The records, originating from different sources and produced at different historical periods, were designed primarily for the reporting of individual events and were not intended to be used in association with each other. Indeed, Macfarlane has suggested that English records have particular characteristics which make them much more difficult to

link than those for other European countries (Macfarlane et al 1983, 20). He cites the fact that individuals are usually poorly identified in the records and that there is little compatibility between the record formats adopted by the State, the Church and the Estates. He concludes, in fact, that record linkage by computer is impossible for the kinds of records which he was using (Macfarlane et al 1983, 29).

3.3.2 Solution

Faced with the problems of inconvenient record formats it is clear that there must be some prior transformation of records into forms which will be more amenable to subsequent processing by the computer. Unfortunately, any process of transformation can degrade the information content of the records, either because the types of transformation are inappropriate (14) or through human error. The effects of human error can normally be minimised by ensuring that the transformations are simple to apply.

By far the most significant transformation is to convert free formatted records into a fixed format style. Considering the three baptism records from the Elwick Hall parish register (1796) which were discussed in the previous section, these could be represented in a fixed format style as follows (15):

B1/7 MARCH 1796/HANNAH/D/THOMAS/FRANCES/AINSLEY/ELWICK HALL

B1/23/JOHN/S/JOHN/FRANCES/SHOTTON/HIGH-BARN/DOB=21

B1/5 APRIL/WILLIAM/S/THOMAS/JANE/WHEATLEY

B1/"/MARY/D/"/"/"

Several comments should be made about this transformation:

1. the information in each record has been disaggregated into discrete items separated by the character '/'.

2. the baptism record for the twins has been transformed into two separate records.

3. the ordering of items in each record is identical, although it is permissible for some records to have missing items, e.g. the third and fourth records have no residence information.

4. each record is prefixed with the special tag 'B1'. This tag serves to indicate the type of record, and each format would have its own unique tag value. When the record is subsequently read into the computer the tag value can be used to invoke the appropriate data validation and processing procedures.

5. a number of facilities are provided to reduce the need for verbose presentation of information. Note the use of the abbreviations 'S', 'D' and 'DOB' for son, daughter and date of birth, respectively. The use of the ditto character (") also obviates the need for entering repeated information. Finally,

for date information the convention is adopted that if the year only or both the month and the year are missing then the corresponding values from the previous entry will be assumed.

Where record formats change radically over time then it is desirable to create formats which match each distinct record style. For example, the two later baptism records (1798-9) could be represented using a 'B2' format as follows:

```
B2/C/THOMAS WHEATLEY/29 APRIL 1798/30/2S
B2/F/THOMAS WHEATLEY/FARMER//GARMONDSWAY MOOR
B2/M/JANE FAWELL/ANTHONY FAWELL/LANCHESTER

B2/C/JANE AINSLEY/15 FEB 1799/5 MARCH/4D
B2/F/THOMAS AINSLEY/HIND/ELWICK HALL/SEDGEFIELD
B2/M/FRANCES FORSTER/JOHN FORSTER/WITTON GILBERT
```

In this case each nominal record is disaggregated into three parts: one providing information about the child ('C'), the second information about the father ('F') and the third information about the mother ('M'). It should be observed that an attempt has been made to match the fixed format style as closely as possible to that of the original records, and the order of presentation of information has been carefully maintained. With formats such as these it is not therefore too difficult to ensure that few errors are introduced when the data is manually transformed from its original source form (16).

Let us turn now to the third problem discussed, viz. the lack of information in nominal records. It is evident that this problem will for the most part be intractable, in that the required information

which is not present in a record cannot be somehow conjured into existence. However, there are some situations where 'new' information can be derived from the information provided. Reference has already been made to the estimation of a person's date of birth when given only their date of baptism. Another example, also already quoted, concerns the possible deduction of a woman's maiden name from information about her relatives given in a census household record. Such inferential techniques as these are valuable, although they do involve computer programming complexity and are necessarily subject to error.

The final problem discussed, viz. the inconsistent presentation of a given type of information, is also not readily solved. From the computer programming viewpoint it is desirable that a given type of information should always be presented in the same way. However, this would mean that information fields would often need to be considerably transformed before input to the computer. Some compromise solutions may therefore need to be adopted, but, in general, the use of standard methods of presentation is to be recommended (17).

3.4 IMPERFECT RECORDING OF INFORMATION

3.4.1 Problem Description

We must finally look in detail at the deficiencies which are implicit in the recording process itself and consider how they may be handled.

Perhaps the most serious deficiency of all is that the records pertaining to a particular individual or family can be geographically dispersed (e.g. in a number of parish registers), and there is no convenient means for locating and bringing them together. The net effect is that record linkage work based on a specific geographical area will necessarily tend to omit details of the lives of the more mobile members of the population. This can unfortunately introduce a bias into any results, and one must take this into account when making demographic calculations (Schofield 1972). For example, one particular study using nineteenth-century Swedish population records (Kalvemark 1977) demonstrates that migrants had a much greater level of upward social mobility than non-migrants.

A second deficiency concerning the recording process is that no source document can be guaranteed to contain a complete record of the people and/or events for which it was ostensibly created. An Anglican parish register, for example, will necessarily omit references to many of the events in the lives of the people of the parish, and

particularly of those, such as nonconformists and Catholics, who would normally have been registered elsewhere (Krause 1965; Razzell 1972; Wrigley 1975; Levine 1977, 153-74). Additionally, even for those who regularly attended the parish church there would be many situations where their events would be missing from the local register. For example, a man who married a woman from some other parish would normally have gone to the bride's church for the marriage: a corresponding marriage record would therefore not have been entered in his own parish register. And, according to custom, the baptism of their first child might also have been celebrated in the bride's church, even at a time when the couple had settled in the groom's parish (Steel 1968, 147).

For those events which took place in the parish church there is still no guarantee that corresponding entries would have found their way into the parish register. For a variety of reasons, such as the absence of the normal incumbent, an entry may have been recorded temporarily onto a sheet of paper, which was later mislaid before the information could be transferred to the register (18).

A third deficiency in the recording process is that events are not always completely reported, i.e. there can be missing information. Thus, for example, there are baptism entries where the name of the child is omitted, and there are census household entries where the person's birthplace does not appear. Such omissions could have arisen because the appropriate information was not available or had been withheld, or alternatively because of some clerical error.

A final major type of deficiency concerns the accuracy of the information contained in the records. In the first place, the person who volunteered the information may not have had the true facts at his disposal: for example, he may not have known his own age and birthplace precisely. But even where the correct information was volunteered the clerk making out the record may have entered it wrongly, either through mis-hearing, carelessness or difficulty with spelling.

3.4.2 Solution

There are no 'solutions', as such, to the problems of imperfect recording of information. If a source document portrays a distorted and patchwork view of reality then it is difficult to see how that view can be 'dedistorted' or made more complete. Even to be in a position to recognise the existence of distortion would imply that one had a standard, 'correct' view against which to make comparisons. In the 'real world' no such standard views exist.

Despite the apparently intractable nature of this problem the situation is not completely hopeless. Since, in general, one will have at one's disposal several source documents, each with its own viewpoint and possible bias, one is able to make comparisons between the sources, and as a result obtain an indication of what distortions are present. This technique is similar to that employed by the police when presented with information from several independent witnesses to

an event. By noting the points of agreement and disagreement among the witnesses the police are able to build up a more precise picture than could be obtained from any single witness.

Let us consider how this strategy may be applied with respect to historical nominal records. It was indicated above that an Anglican parish register would tend to under-represent certain groups in the population, such as nonconformists and Catholics. One feasible way to investigate such a phenomenon, at least in the nineteenth century, is to attempt to cross-link the parish register records with some alternative source, such as census records (Razzell 1972; Wrigley 1975; Levine 1977, 156-68). By observing discrepancies and missing links between these it is then possible to estimate the degree of under-representativeness in the parish register data (19).

The problem of missing information in records is also not easily dealt with. In some cases the record linkage process can, as it were, 'fill in' the information. For example, where the birthplace information is missing in a census record it may be possible to obtain the appropriate information from a second census record to which the first one will link. However, the very fact that information is missing may on occasion render record linkage impossible, particularly where it is the name information which is not present.

Finally, there is the problem of inaccurate information. Methods for dealing with variations in names were considered in Sections 3.1.2 and 3.2.2. A similar flexibility is required with respect to age and

date of birth information. For example, it is conceivable that the age given in a census record could be in error by one or two years (20), and a linkage system should be designed to tolerate such a discrepancy. However, where errors are caused by careless reporting, such as where an age is represented as '34' instead of '43', then it would be unreasonable to expect the system to be able to exercise the same kind of flexibility in a consistent fashion.

NOTES

1. There is evidence to suggest, however, that although nominal ambiguity may be a difficult problem to handle it occurs in practice rather less frequently than is sometimes imagined. Levine, for example, in his work involving the manual linkage of English pre-industrial parish register records (Levine 1977, 155) comments that 'the process of reconstituting families was not particularly difficult'. Indeed, in less than 1% of the 25,000 entries which he linked did he have any doubt about the validity of the link which was made.

2. For a discussion of the application of demographic tests see Wrigley and Schofield 1973, 73-5.

3. But it should be noted that this calculation will not normally be valid for adult ages reported in the 1841 census. Here the age was intended to be rounded down to the nearest multiple of five years. Thus, all those in the 35-39 age-range were to be recorded as aged 35. Likewise, the calculation may not be valid for an age in a marriage record which is given as '21': in such a case it may merely have indicated that the person was over 21 years of age.

4. The Driver Number on a UK Driving Licence is just such an identifier. Each 16-character code contains the person's date of birth, sex, initials and the first five letters of his surname, together with two characters used for computer error-checking.

5. The standard algorithm for constructing Soundex codes is described in Winchester 1970, 115. The basis of the code is that each surname is converted into a code value consisting of a letter followed by three numerals. The first letter of the surname is used as the initial letter. Subsequent vowels and the letters Y, W and H are ignored. The first three of the remaining consonants in the surname are coded phonetically. For example, the letters B, P, F and V are all given the code value 1. An evaluation of the Soundex scheme and some of its variants is provided in Wrigley and Schofield 1973, 98-101.

6. There have been several refinements to the basic Soundex scheme, in order, for example, to improve the handling of spelling idiosyncracies in foreign names. Another type of refinement is to incorporate mechanisms which take account not only of the aural similarity of names but also their visual similarity: such enhanced systems are sometimes referred to as 'viewex'. For an examination of these refinements see Hershberg et al 1976A, 141-3.

7. It is also possible to develop a hybrid scheme which makes use of both algorithmic and directory techniques (Bouchard and Pouyez 1980).

8. The following examples of incorrectly ascribed relationships in 1851 census data are quoted in Anderson 1972, 76: brother-in-law described as son-in-law, son-in-law described as step-son, father-in-law described as step-father, and niece described as sister-in-law.

9. The relevant analyses for Elwick Hall parish, presented in Section 12.3.4, reveal a significant discrepancy in the methods of reporting occupations in the census and parish register records. The conclusion reached is that any study of occupational mobility which makes use of disparate source materials must be approached with particular caution.

10. Strictly 'ON', indicating occupation-name, would be a more appropriate tag than 'JN'. However, in practical computing terms the risk of having the 'O' in 'ON' mistyped as the number '0' has weighed against adopting this alternative.

11. A significant advantage of this simple scheme is that it is consistent with the method of handling other name information (Christian names, surnames, etc.). This has two beneficial consequences. Firstly, the user interface is kept simple and coherent. And secondly, because it is possible to use common mechanisms to handle the various kinds of name information this solution is simpler to implement.

12. Macfarlane has attempted to handle such amorphous records for the parish of Earls Colne in Essex. However, the method which he adopted did not rely solely on the use of sophisticated programming techniques. Rather, all records were manually 'preprocessed' into a form which was more manageable for the computer. This involved the imposition of a structure on the records by the inclusion of special syntactic marker and bracketing characters where appropriate. Jardine and Macfarlane 1978, 76-7.

13. DCRO 1978 Vol 2, 16. This highly detailed form of baptism entry was adopted by all the parishes in the Durham Diocese during the period 1798-1812. A similarly extended form was also introduced for burials. Although the transcription here suggests a free format style entry, columns were in fact ruled in the Elwick Hall register for: Name, Birth, Baptism, Child and Names of the Parents. The use of a free format style here is merely for the convenience of presentation.

14. A particular type of transformation which is not recommended is that involving manual 'pre-coding': for example, replacing occupational titles by numeric code values. The main disadvantage of this approach is that it usually involves the convergence of several name strings (e.g. occupational titles) into one numeric code value. As a result, the informational content of the machine-readable version of the data is irrevocably degraded. For an examination of these issues see Floud 1979, 207-10.

15. The fixed format styles illustrated in this section and used in the present work are similar to those employed by the Cambridge Group (Schofield and Davies 1974) and by Anderson's system for handling 1851 census data (Anderson et al 1977).

16. Data validation procedures should detect most of the remaining errors. An additional strategy is to have two people enter the same data independently: any discrepancies between the two versions can then be examined and appropriate, corrective action taken. An alternative, more recent approach to data entry involves the use of a portable microcomputer, which is able to validate the individual data fields on entry and map them into the required card image formats (Olsen 1985).

17. The following represent the main standards of presentation which have been adopted in the present research:
    - dates are expressed in the form 'day month year', e.g. '4 NOVEMBER 1868', but the month may be abbreviated to its initial three characters.
    - ages are normally expressed in the form of an integral number of years, e.g. '28'. Where an age is to be represented in terms of some other unit, i.e. months, weeks, days or hours, then the number is followed by a space and then a single character code indicating the unit. Thus, an age of seven months would be expressed as '7 M'. The codes for weeks, days and hours are 'W', 'D' and 'H' respectively. Where an age is expressed in terms of some imprecise designation, e.g. 'infant' or 'minor', then such a designation (or an approved abbreviation) is allowed.
    - names are expressed precisely as they appear in the records, except that titles are omitted. If it is desired to retain a title, or indeed any other information, in the source record but have the system ignore it then it may be placed within angle brackets. For example, the name 'Sir James Brown' should be entered as '<SIR>JAMES BROWN'.
    - marital status can be expressed in a wide variety of ways, including: SINGLE, UNMARRIED, BACHELOR, SPINSTER, MARRIED, WIDOW and WIDOWER, together with abbreviations.
    - placenames are expressed precisely as they appear in the records, except that where a multi-level address is given, the individual parts of the address are separated by commas (,). For example, 'Dovecot House in the Parish of Hart' is expressed as 'DOVECOT HOUSE,PARISH OF HART'. This enforced subdivision simplifies the subsequent handling of the address information.
    - occupations are expressed precisely as they appear in the records, except that additional information and qualifications may be expressed in a formalised way. For example, 'Master Mariner' may be expressed as 'MARINER,STATUS=MASTER' and 'Retired Butcher' as 'BUTCHER,STATUS=RETIRED'. Further, the detailed census description 'Farmer of 212 acres, employing 3 men and 2 boys' can be expressed in a formalised way as:

'FARMER,ACRES=212,MEN=3,BOYS=2'.
And finally a dual occupation, such as 'Blacksmith and Publican', may be presented instead as 'BLACKSMITH,PUBLICAN'. Such a method of organising occupational information simplifies its subsequent handling. It also helps to ensure that the directory of occupational titles is kept to a modest size, whilst at the same time avoiding any loss of information.

18. Evidence of such practices is provided in Steel 1968, 27-32. Indeed, many parishes appear to have deliberately adopted the use of a memorandum book into which the original entries would be written. Such entries would then be transferred to the parish register at some convenient opportunity, perhaps once a year. Steel cites the following example of note-keeping from the parish register of St. Peter's, Dorchester:
    1645. In twelve months there died 52 persons whose names are not inserted, the old clark being dead who had the notes.

19. For Colyton in Devon, where he linked the Anglican baptism register to the 1851 census, Wrigley discovered that 79.1% of those claiming in the census that Colyton was their birthplace had entries in the Colyton baptism register (Wrigley 1975). Of the remainder, 6.3% were found in two local nonconformist registers, 7.5% were found in neighbouring Anglican registers and 7.2% could not be found. (N.B. This analysis excluded a number of married and widowed women whose maiden names were not known.)

20. See Chapter 10, Note 9.

CHAPTER 4 DEALING WITH THE 'REAL WORLD'

Having examined the problems of nominal record linkage in depth it is now necessary for me to address issues on the opposite side of the interdisciplinary divide, viz. the nature of the computing 'mechanisms' which may help to achieve a solution. There are two crucial issues which must be addressed when using computing facilities. The first concerns the selection of the best facilities for the task in hand. The second concerns the adoption of the most appropriate ways of using these facilities. My approach to these issues in this chapter will be a general and introductory one. In subsequent chapters I shall look more closely at how the computing facilities may actually be used in the context of the particular problem with which I am concerned.

A fundamental characteristic of nominal record linkage is its complexity. A vital requirement, therefore, is that particular attention should be given to methods of handling complexity, and this will be addressed in Section 4.1. Later in the chapter I shall examine the role which a database management system can play in simplifying the organisation of data, and shall look, in particular, at the way in which the IDMS system can be used to model complex genealogical structures.

4.1 SOLVING THE PROBLEMS OF COMPLEXITY

It has been said that 'the art of programming is the art of organising complexity'. And this adage gains more significance as attempts are made to tackle increasingly complicated problems with computers. How then should one proceed with the development of computer systems without being overwhelmed by complexity and the myriad of technical problems which can arise? Three main lines of approach can be identified, as follows:

4.1.1 A Strategy of Disaggregation

It is generally understood that the task of the computer programmer involves taking a difficult problem and breaking it down into a number of simpler problems, which are themselves broken down, and so on, until eventually one is left with a sequence of elementary operations. The programmer's task, in other words, is to disaggregate from the difficult down to the simple. But it should be noted that this strategy of disaggregation is not restricted merely to the programming task itself: it can be used as a guiding principle throughout the entire development of a computer system.

An example of the way in which the strategy can be usefully applied is in the planned subdivision of the total development into a series of discrete stages. Whereas one could adopt a monolithic approach, and aim to build a complete system 'at one go', the

alternative and preferred approach is to set out to build a primitive system initially, and then to extend it progressively with additional facilities. Thus, according to computer consultant Tom Gilb (1981):

> I suggest that we adopt a different mentality towards system plans. We should recognise that it is often in our interest to divide up a development process into many small steps of progress. And we should applaud the designers and planners who manage to identify relevant steps. I believe that they are demonstrating greater imagination and talent in their work, while at the same time showing maturity and caution in not trying to accomplish too much at once.

The advantages of adopting such a phased development are as follows:

1. it allows for more design flexibility. Since one is always working towards limited goals one can often delay making decisions about particular aspects of the design until a much later stage. With a monolithic approach one is constrained to make all the important design decisions at the start, and the inevitable result is that one becomes inflexibly 'locked in' to the original design 'view'.

2. it enables one to guarantee that a system, albeit a primitive one, will ultimately be produced. With a monolithic approach it is difficult to offer guarantees of any kind.

3. it recognises that design and implementation operate in an iterative fashion. Thus, for example, since a phased approach enables one to develop and make use of primitive versions of the system, this experience provides valuable feedback which

can guide the subsequent design work.

4. it enables one to give priorities where they are seen to be desirable. For example, one can decide that a certain group of facilities is so important that it should be included in the first, primitive release, whereas another group of facilities is much less important and should be incorporated at a much later stage, and only then if time permits.

5. the resulting system is likely to be more error-free and reliable. This is so, firstly because the system will have been assembled in a progressive and controlled manner, with at each stage the main objective being that the system should function properly and according to its specification. But secondly, since the system will have been in operation over a prolonged period there will have been ample opportunity for the detection and correction of most errors.

A second example of the value of employing a strategy of disaggregation concerns the preparation of data. Ultimately, when a system becomes operational it will be required to process large quantities of data. However, it is vital that the task of preparing this data should not be embarked on prematurely in the timescale of the project. The design of the data input formats is an integral part of the design of the system, and for maximum flexibility, therefore, it is crucial that during the early stages only a small amount of data should be prepared, just sufficient for program testing purposes.

Should it subsequently be decided that modifications to the formats are needed, the task of editing or repreparing the data will then be minimised. A phased approach to the preparation of the data should therefore proceed in parallel with a phased approach to the development of the system.

A strategy of disaggregation has been employed in the present design, and the first operational linkage system had a quite limited range of capabilities, as follows:

1. it handled only one type of nominal record, viz. census household records (1851, '61 and '71).

2. it carried out person-based linkage only, i.e. it created links only between people and the records in which they were identified. The development of family-based linkage facilities and person-to-person links was incorporated later.

3. it provided only simple analytical facilities, these being designed primarily to display the results of the linkage so that the validity of the linkage strategies could be monitored.

A final example of the use made of disaggregative and evolutionary development strategies concerns the selection of a study population. In order to minimise the initial data preparation requirements it was decided to use in the first instance data only for

the small parish of Elwick Hall (population c. 200). It was intended that, time permitting, the larger, neighbouring parish of Hart (population c. 750) would also be incorporated at a later stage. While in the event the timescale did not permit the incorporation of this second parish, the employment of a disaggregative development strategy ensured that the project could nevertheless be brought to a successful conclusion.

4.1.2 Simplifying the Organisation of Data

The linkage of nominal records is a complex process, involving the interweaving of records by a network of interconnections: as such it represents a difficult computer programming problem. This problem consists of two sub-problems. Firstly, how should complex data structures be organised? And secondly, how should one implement programming strategies to manipulate such complex data structures? I shall attempt to answer these questions in the present and following sections.

It was remarked in Chapter 1 that the introduction of database technology was to be viewed as an event of crucial relevance to the solution of record linkage problems. Since this development has provided a powerful means of reducing complexity, particularly in relation to the organisation of data, it is appropriate that I should now address attention to the nature of this development.

It will be instructive to begin by reviewing briefly the history of computer data storage. The earliest computers had no facility for storing and accessing large quantities of data: they were limited by the capacity of their main-store memory. As a result, the earliest computer applications were restricted mainly to mathematical calculations, normally with the data being input and results output via the medium of paper-tape.

With the subsequent introduction of magnetic storage media, such as tapes and discs, the way was open for a whole range of new kinds of applications. In addition to acting as a computational device the computer could now become an information storage device, an 'electronic filing cabinet'. Any organisation with a computer would now be able to use it to hold its files of business information: such files could then be regularly updated and used to print payslips, requisitions, invoices, etc. Gradually the computer was beginning to reflect or model the organisation that was using it. For example, in the 'real world' of the organisation there might be departments for dealing with wages, invoices and stores, while in the computer system there would be corresponding suites of programs for handling payroll, invoicing and stock control, respectively.

However, there was one important respect in which, at least with conventional filing systems, the computer did not adequately model the 'real world' of the organisation. Within the computer system each application would usually operate in complete isolation from all other applications, and it would maintain its own private set of files. For

example, the payroll program would have a set of tapes which it would use at regular intervals, but which would never be used by any other program. Clearly, such fragmentation of the organisation's data was not a true reflection of the way that the organisation would in fact be structured and run. In reality each department would need to have some dealings with the others, and any major organisational objective would require the close collaboration of several departments. For many purposes the mismatch between the 'computer view' and 'reality' would not greatly matter, but with increasing automation and reliance on the computer it was likely that more and more problems would emerge.

To take a trivial example of the kinds of problems that could arise, let us suppose that a customer were to inform an organisation that he had changed his address. It is conceivable that the address information might be maintained by three or four separate applications, and therefore the change of address would need to be communicated to each of these. Failure to do this could lead to the kind of embarrassing situation where goods were being delivered to the old address while invoices were arriving promptly at the new address! And it is to be observed that the root cause of this problem lies in the unfortunate mismatch between 'reality' and the 'computer view'. In the 'real world' the customer would normally have one address; but inside the computer he could have several.

The solution to this mismatching problem requires the abolition of isolated files of information and their replacement by a shared

body of data which is to be used by all applications. In this modified environment each customer would, for example, have his address stored only once, and therefore a change of address would simultaneously become effective for each and every application. This setting up of a shared body of data is referred to in computing terms as the setting up of a 'database', and at present it represents the most elaborate way of organising computer data storage.

But precisely what significance does the setting up of a database have for nominal record linkage, and, in particular, how can it simplify the organisation of data? In the above example it was demonstrated that traditional, fragmented filing systems can introduce a mismatch between the 'computer view' and 'reality', and that the adoption of a database approach can serve to reduce this mismatch. Furthermore, it should be emphasized that the customer address problem arises, not because it is a significant problem in the 'real world', but rather because the computer's total model of 'reality' is faulty. The main advantage therefore to be gained from the adoption of a database approach is that it enables us to avoid the problems which arise through inferior computer modelling and leaves us only with the problems which are intrinsically part of the 'real world' with which we need to be concerned. In Sections 4.2 and 4.3 I shall explore further the ways in which database management systems can assist the accurate modelling of the 'real world' and thereby simplify the organisation of data.

4.1.3 Simplifying the Organisation of Programs

It would be naive to assume that merely by using a database system all difficulties will disappear. If our 'real world' problem and the associated data structures are complicated then it is inevitable that we shall be faced with a considerable programming task. How then should we approach this task?

Some introductory comments were made in Section 1.5 about the nature of computer problem-solving, and an important observation was made concerning the iterative nature of the total process. In particular, it was noted that problem analysis and solution implementation are not normally discrete processes which proceed one after the other. But, rather, that it is more likely that they will work in parallel, with the implementation of a solution leading to an increased understanding of the nature of the problem, and so on. In the following discussion, therefore, where the main elements of the programming task are examined, it is crucial that the underlying, iterative nature of the total process should be continually borne in mind.

A fundamental element of the programming task, but one which is sometimes neglected, is problem analysis. The danger in all programming work is that by not giving sufficient attention to problem definition one may subsequently produce a program which, even if it works, does not fulfil the requirements placed upon it.

Having analysed the problem it is then necessary to decide what it is that one is setting out to achieve. It is at this stage that one is required to make selections from all the options available. For example, is the computer system to be capable of handling all kinds of nominal records, or is it to be special-purpose? Will an attempt be made to process records in their natural source form, or will they be pre-formatted in some way? Is the system to be capable of handling very large populations, or will there be size limitations? Will the linkage process be required to operate fully automatically, or is there to be provision for human intervention and decision-making? Are there plans to write programs for organising the storage and retrieval of records, or will use be made of a proprietary database system? How such questions are answered will to a large extent determine the magnitude of one's task and whether or not the project will be a feasible one.

The next task is to plan a strategy for achieving the design objectives, and a strong recommendation has already been made in Section 4.1.1 for the subdivision of the total development into a series of discrete stages. As far as possible the strategy should permit a number of iterations of the 'problem analysis-solution implementation' cycle.

The next major stage is the detailed design of the computing system. Having established what the system should be capable of doing the task is now to decide how it should do it. A comprehensive examination of the methods of designing and writing programs does not

properly fall within the scope of this thesis. Indeed, it is a large subject. However, it may be worthwhile at this point to examine some of the main principles.

A primary concern in programming is that programs should 'always be simple, even if the tasks they perform are complex' (Jackson 1975, v). Such simplification can be achieved at two distinct levels. The first concerns how the complete program is subdivided into its functional parts; the second is concerned with how the individual parts of the program are actually written using the statements of a particular programming language.

There is no agreed 'best method' for deciding how a program should be divided into its functional parts. Many people favour a predominantly 'top-down' approach, in which one starts with the total problem to be solved and then progressively disaggregates it until one is left with the specification of a large number of relatively simple routines which can then be coded. Alternatively, one can adopt a 'bottom-up' approach, where one starts by designing and coding a number of primitive routines which carry out tasks which are seen to be necessary in the overall scheme. Progressively the primitive routines are assembled together to create programs of increasing capability and complexity, until ultimately one has a program which can satisfy all the functional demands placed upon it.

Michael Jackson is an exponent of the 'top-down' approach, and he has devised an elegant method for designing programs, which is

particularly appropriate for database-oriented problems. The essence of his method is that the program structure should precisely match the data structure. He has also designed a simple but effective diagrammatic notation for representing data structures. The use of this notation enables complex data structures to be disaggregated and simplified, and this process simultaneously produces the overall design of programs which are needed to handle the data (Jackson 1975, 15-38).

To take a simple example of the Jackson approach, let us suppose that it is necessary to design a program to handle population census enumerators' records. Such data has an explicit hierarchical or tree-like structure, in that each enumeration district has many households in it, and each household has many occupants in it. Jackson would therefore recommend that the appropriate structure for the program would also be hierarchical, consisting of, perhaps, three routines, one to process an enumeration district, the second to process a household and the third to process an occupant. The top-level 'ENUMERATION DISTRICT' routine would be responsible for handling the information associated with one enumeration district, such as its serial number: it would also be responsible for repeatedly calling the 'HOUSEHOLD' routine until all households in the enumeration district had been processed. Similarly the 'HOUSEHOLD' routine would be responsible for handling the information associated with one household, such as its address and schedule number: it would also be responsible for repeatedly calling the 'OCCUPANT' routine until all occupants in the household had been processed. Finally, the

'OCCUPANT' routine would be responsible for handling all the information associated with one occupant, e.g. name, age, occupation, etc.

The principal advantage of this kind of functional subdivision is that the resulting program has a structure which faithfully reflects the problem or 'real world' structure. Thus, just as in the 'real world' there are enumeration districts, households and occupants, so in the program there are corresponding and distinct sections of code to deal with these items. Should it become necessary for the program to be enhanced to deal with a slightly modified 'real world' problem then it is a relatively simple task to locate the precise position in the program where changes will need to be made. Where programs have a structure which bears little relation to the 'real world' structure the task of making modifications can be significantly more difficult.

Let us turn now to the second level at which it is possible to make programs 'simple'. This concerns the method of actually writing the program statements. If we were to examine a selection of program code samples we would probably find that some of these were easy to understand, while others were difficult. But what, surprisingly, we might also find was that the ease or difficulty of understanding was not at all related to the complexity of the problems being handled. Instead it would probably be the style of programming which was the determining factor.

What then are the characteristics of a good programming style, one which is conducive to the production of program code which is 'simple' and easy to understand? This subject is covered widely in the literature, and the favoured style of programming is normally referred to as 'structured programming' (1). The main tenets of this approach are as follows:

a. the structure of each functional unit of code (e.g. a module or routine) should reflect the structure of the problem with which the unit is concerned.

b. each section of code should be constructed from only three basic structural element types, as follows:
    1. sequence - where control always passes from one sub-element of the sequence to the next.
    2. iteration - where all sub-elements of the iteration are repeatedly executed until a particular condition is reached.
    3. selection - where, depending on the value of one or more variables, one section of code will be executed in preference to another or other sections of code.

c. the three basic structural element types may be nested indefinitely within each other. For example, a module might typically consist of a sequence of 3 sub-elements, the first two being simple assignment statements and the third an iteration. The iteration might itself contain a further

sequence, sub-elements of which could be further iterations and selections, and so on. An example of this kind of statement nesting is provided in Section 5.2.3.

d. the 'GO TO' statement should be avoided, as far as possible.

The overriding objective of this approach is that control should always pass in a highly disciplined way from one section of code to the next, and the stricture about the use of the 'GO TO' statement is intended to facilitate this. The alternative approach to programming, where the 'GO TO' statement is used freely and in a rather arbitrary way to transfer control haphazardly about the code, can lead to programs which are difficult to understand and which are more likely to contain logical errors.

4.2 AN INTRODUCTION TO DATABASE MANAGEMENT SYSTEMS

It was suggested earlier in this chapter that the main advantage of adopting a database approach was that it would help us to refine the computer's total model of 'reality' and bring it more into line with the 'real world'. It was also suggested that there would be a consequent simplification in the organisation of data. It is necessary now to consider how in practical terms this can be achieved with the aid of a database management system.

The concept of a database system as a provider of highly sophisticated filing facilities is portrayed graphically as follows (Martin 1976, 3):

> The use of a data base is like having a superbly fast and brilliant Dickensian clerk who keeps data for many applications. He organizes his books so that minimum writing is necessary and so that he can search the books quickly to answer any queries that may come along. Unlike his pedestrian predecessors who could write or read items in only one ledger, he rushes from one set of data to another collecting together separate items to respond to highly varied requests for information. He is a godsend to management.

At present, database concepts are still evolving, and there is a range of alternative facilities available. For our present purpose it will be worthwhile to examine the three most commonly occurring types of database systems, viz. the hierarchical, network and relational systems. Each of these is designed to handle 'real world' problems with differing characteristics. Eventually it is to be expected that there will be a convergence of these separate approaches and the provision of a more unified set of facilities.

4.2.1 Hierarchical Database Systems

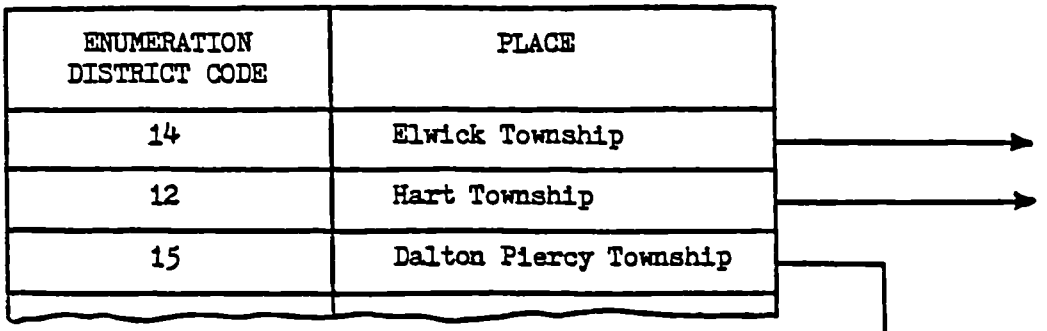
The simplest kind of data organisation is one where the information is stored in isolated 'flat' files. In such a file all the records are of the same size and structure. A payroll file which contains one salary record for each employee in a firm is an example of such a flat file: this is illustrated in Figure 4.1.

NAME	PERSONNEL NUMBER	SALARY (£)	TAX CODE
GRAY, John	125635	5200	145L
BAKER, Rachel	091927	3500	137L
WARDLAW, Tom	071078	3900	274H
MITCHELL, James	065529	6800	196L

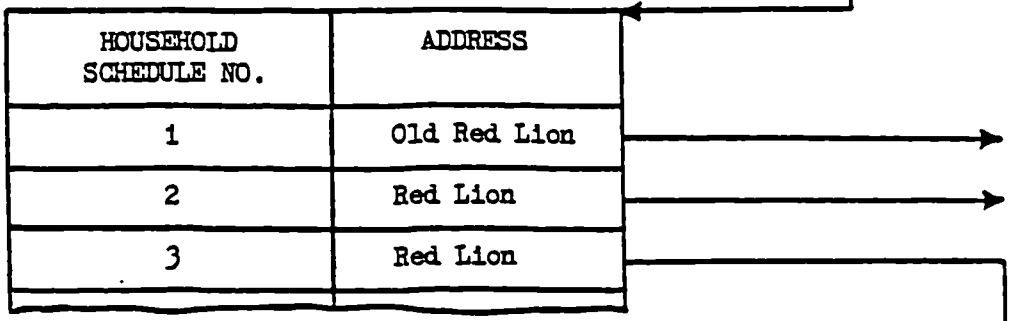
Figure 4.1 An example of a flat file

In general, data does not fit naturally into a flat file type of organisation, and so other, more complex, types of organisations need to be considered. One of these is the hierarchical or tree-structured organisation. Reference has already been made in Section 4.1.3 to the hierarchical nature of population census enumerators' records. Unlike the payroll file, where only one type of entity is being described, viz. the employee, a population census file contains information about three basic entities: enumeration districts, households and occupants. Such a file therefore needs to contain three different kinds of records, each with its own size and structure, to correspond with these three entities. An example of such a file is illustrated in Figure 4.2. A heterogeneous organisation of records such as this can no longer be described as a flat file, and moderately sophisticated computing facilities are needed to access it.

In a flat file each record will normally be treated as an isolated entity, and as a result the file processing operations are relatively simple. Thus, a programmer would be able to gain access to such files merely by incorporating appropriate 'READ RECORD' and 'WRITE RECORD' statements in his programs. But in a hierarchical organisation the records will have implicit relationships with each other, and therefore processing is likely to be much more complicated. In a population census file, for example, one particular group of occupant records will be associated with one of the household records, another group with another household record, and so on. The user must therefore be able to employ some method of associating or linking the records together. If he so wishes he can design and code his own



ENUMERATION DISTRICT CODE	PLACE
14	Elwick Township
12	Hart Township
15	Dalton Piercy Township



HOUSEHOLD SCHEDULE NO.	ADDRESS
1	Old Red Lion
2	Red Lion
3	Red Lion

OCCUPANT NAME	RELATION	CONDITION	AGE		
			MALE	FEM.	
Thomas HODGSON	Head	Mar	50		etc.
Jane HODGSON	Wife	Mar		44	
Charlotte Ann HODGSON	Daur	Unm		21	
Thomas HODGSON	Son		14		

Figure 4.2 An example of a hierarchical file

special-purpose file-handling routines to carry out the required operations. Alternatively, and certainly simpler from his point of view, the user can make use of a package of general-purpose file-handling routines, which will effect the required operations on his behalf. GENDATA is a record-handling system of this kind (Schofield and Davies 1974), and there are database systems which provide comparable facilities.

IMS (Information Management System) is an IBM database system which is designed to handle hierarchical data (2). Such a database system has responsibility for organising the physical storage and retrieval of records, and for associating the records in whatever way is required. A user wishing to access the database achieves this by making requests to IMS, either from a program which he writes himself, or via a purpose-built query program, which is normally used interactively at a computer terminal. In order to examine the nature of the facilities provided by IMS I shall consider the former type of access only. From within his program the user does not need to be concerned with the potentially complex physical organisation of the data. For example, if population census data were stored in the database and the user wished to obtain a particular household record he would merely need to issue a 'GET UNIQUE' request, specifying which household was required, for example, via its enumeration district and schedule numbers. IMS would then carry out the necessary actions to retrieve the record. If the user next wished to obtain all the occupant records for this particular household he would repeatedly issue the 'GET NEXT WITHIN PARENT' request, and IMS would furnish the

required records, finally replying with the special 'NOT FOUND' return code when all occupant records had been retrieved.

In effect then the user has at his disposal a whole range of database commands (collectively called a 'data manipulation language') which enable him to gain access to the data which he requires (Date 1986, 513-20). Provided with such commands the user is said to be able to operate at a 'logical' level, dealing only with a simple, non-physical view of the data, while the database system operates at the 'physical' level, where it needs to be concerned with storage devices, record pointers and the complex organisation of the data. Such facilities can clearly provide the user with a simplified means of organising and accessing his data.

4.2.2 Network Database Systems

In essence network and hierarchical database systems have much in common. They are both systems for organising records of different types, and which the user wishes to associate in a particular way to model a 'real world' view. Both types of systems also provide a convenient 'logical' level at which the user can operate and a corresponding data manipulation language. Where they differ is in the types of structures which they can handle.

In the previous section population census data was presented as an example of hierarchical data. Unfortunately, it is not so easy to

provide examples of network data, since such data does not often occur naturally, at least in a readily tangible form. A rather simple network structure is, however, exemplified in a library index. This essentially consists of two intersecting hierarchical structures, as follows:

1. a subject hierarchy. The most notable example is the Dewey Decimal system of classification, a vast tree-structure in which the major branches of knowledge, such as Pure Science, Social Sciences, Arts, etc., subdivide into sub-branches of knowledge, which themselves subdivide, and so on. For example, Pure Science subdivides into Chemistry, Physics, etc. Chemistry then subdivides into Inorganic Chemistry, Organic Chemistry, etc. Inorganic Chemistry subdivides into Gases, Metals, Non-Metals, etc., and so on. Ultimately, at the ends of the branches are the classification code values which are used to identify the books in each subject area.

2. an author index. This is a somewhat simpler hierarchy, in which there is a collection of author names held in alphabetical order, and for each name there are references to the one or more book titles corresponding to that author name.

Each book in the library can therefore be located via its position in the subject hierarchy or via its author name. Such a structure is described as a network, and a simple example is illustrated in Figure 4.3. But what, one may ask, is the property of

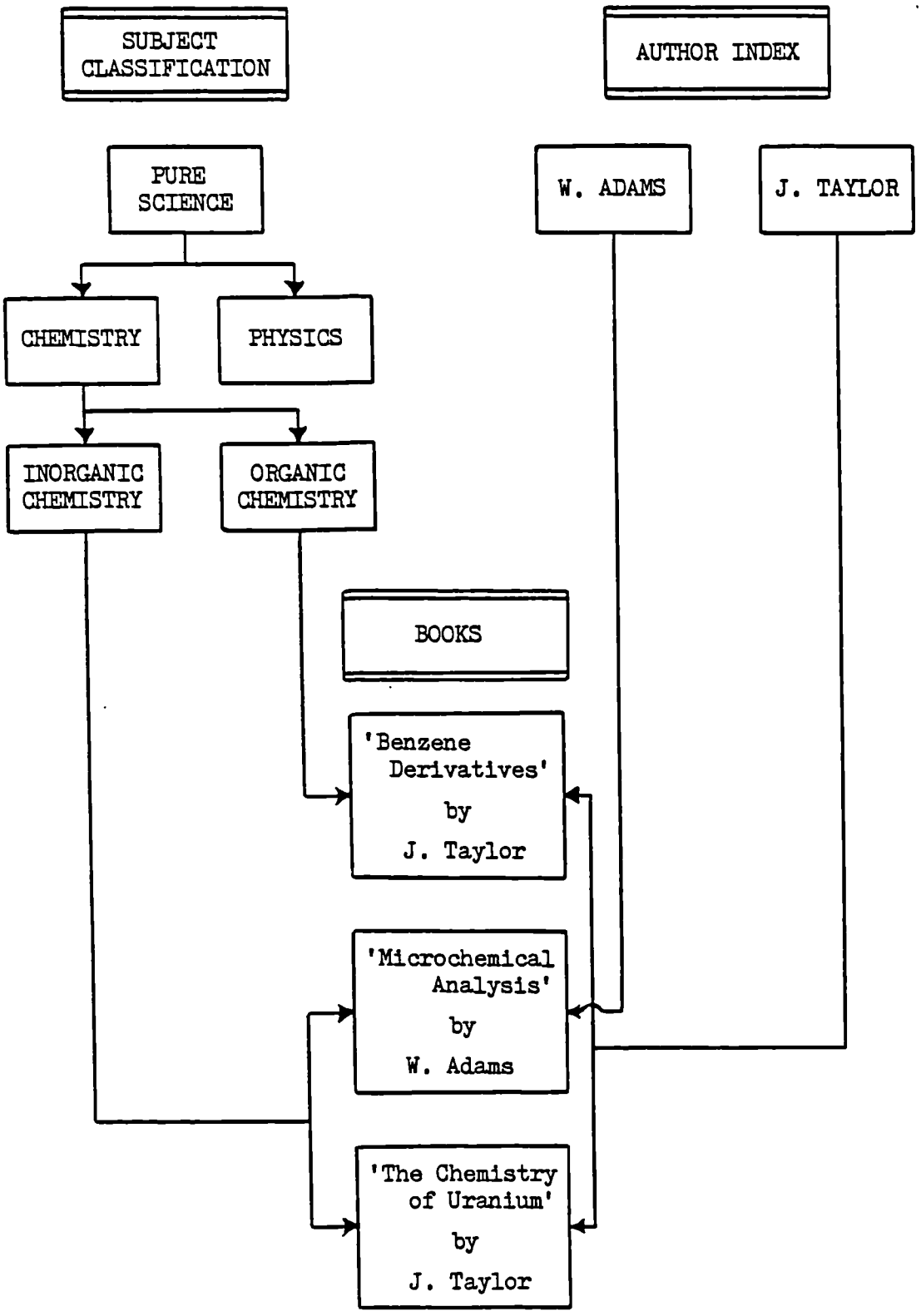


Figure 4.3 An example of a network structure

this structure which distinguishes it as a network, rather than a hierarchy? Quite simply, if one takes any two positions (or 'nodes') in a network then there can be several routes connecting them. For example, if one takes the nodes 'CHEMISTRY' and 'J. TAYLOR' in Figure 4.3 then there are indeed two routes connecting them, one via the book 'The Chemistry of Uranium' and the other via the book 'Benzene Derivatives'. By way of contrast, in a hierarchy there can be only one route between any two nodes. Potentially, therefore, network structures can be considerably more complicated than hierarchical structures, and so they are capable of modelling more complex 'real world' views.

The IDMS database system is an example of a system designed to handle network structures. An introduction to IDMS and an examination of the way in which it can be used to model population structures are provided in Section 4.3.

4.2.3 Relational Database Systems

Historically, the earliest database systems to be used widely were of the hierarchical and network types. However, increasingly, use has been made of a third type, viz. the relational database system.

An important disadvantage of the non-relational systems is that the view of 'reality' which they model is effectively 'frozen into'

the user's programs. Should it be required to modify the view then considerable reprogramming may be necessary, and reprocessing of the entire contents of the database may also be required. The intention with relational systems is that the data in the database should have much more 'malleable' properties, and that the user should at any time be able to adopt any particular 'view' of the data that he desires. This characteristic is described as 'program-data independence'.

In order to provide this kind of flexibility relational systems impose strict rules on how the various information fields may be grouped within the different record types (3). However, once these requirements have been satisfied the data is then amenable to highly sophisticated methods of access. For example, with population data one could dynamically request the database system to locate all occurrences of a particular type of individual, e.g. all married farmers over 45 years of age, and then ask it to provide specific information about each of them, such as their household address and birthplace.

Despite the obvious attractions of the relational database approach it must be said that there can be disadvantages in using relational systems for certain purposes. In the first place the fundamental unit of data manipulation in a relational system is the 'table', rather than the individual record. Applications which are more concerned with individual records than with groups of records may therefore not be completely suited to the relational approach. For example, the process of nominal record linkage itself is intimately

concerned with the connecting of isolated records into networks, these networks being made up of individual hierarchies of records extending to variable depths. It is clear that the relational, 'tabular' view does not closely reflect this orientation.

A second disadvantage with relational systems is that their underlying technology is newer and so somewhat less mature and stable than alternative network technology.

A final disadvantage with relational systems concerns their operational efficiency. The ability of such systems to provide the user with a more flexible interface to his data is achieved at the expense of reduced processing efficiency. Unless, therefore, the improvement in flexibility is an essential requirement a user may prefer to opt for the greater efficiency which can be obtained from a non-relational system.

4.3 AN INTRODUCTION TO IDMS

The IDMS database management system was selected for use in the current research chiefly because of its ability to model population structures. (A more closely argued justification for the choice is provided in Section 5.1.2.) It is appropriate at this point to introduce its main concepts and philosophy. IDMS is a CODASYL-type system, which means that its design is based on the standards

recommended by the U. S. CODASYL Committee (4). In the following sections the main modelling features of IDMS are introduced and the characteristics of a population database are progressively developed.

4.3.1 The CALC Concept

An important feature of a database system is that it should provide efficient access to information. For example, if one had a database containing 1 million birth records then one might reasonably expect to be able to locate any one record without the need for a lengthy search through many thousands of records. This requirement is handled in IDMS by the CALC facility.

Figure 4.4 shows how the CALC facility might operate, with specific reference to the birth record for 'JOHN SMITH'. When the record is initially presented to IDMS for storage the name field, 'JOHN SMITH', is obtained from the record and is submitted to a Hashing Algorithm routine within IDMS. The function of this routine is to do a 'CALC' (i.e. a CALCulation) on the characters of the name, and generate some number as a result. In the example shown it is supposed that the routine translates 'JOHN SMITH' into '6184'. IDMS then uses this number as a storage address, and so would store the 'JOHN SMITH' record at address 6184 on its storage device. Each other birth record would be treated similarly, and would be stored at an address derived from the contents of its name field.

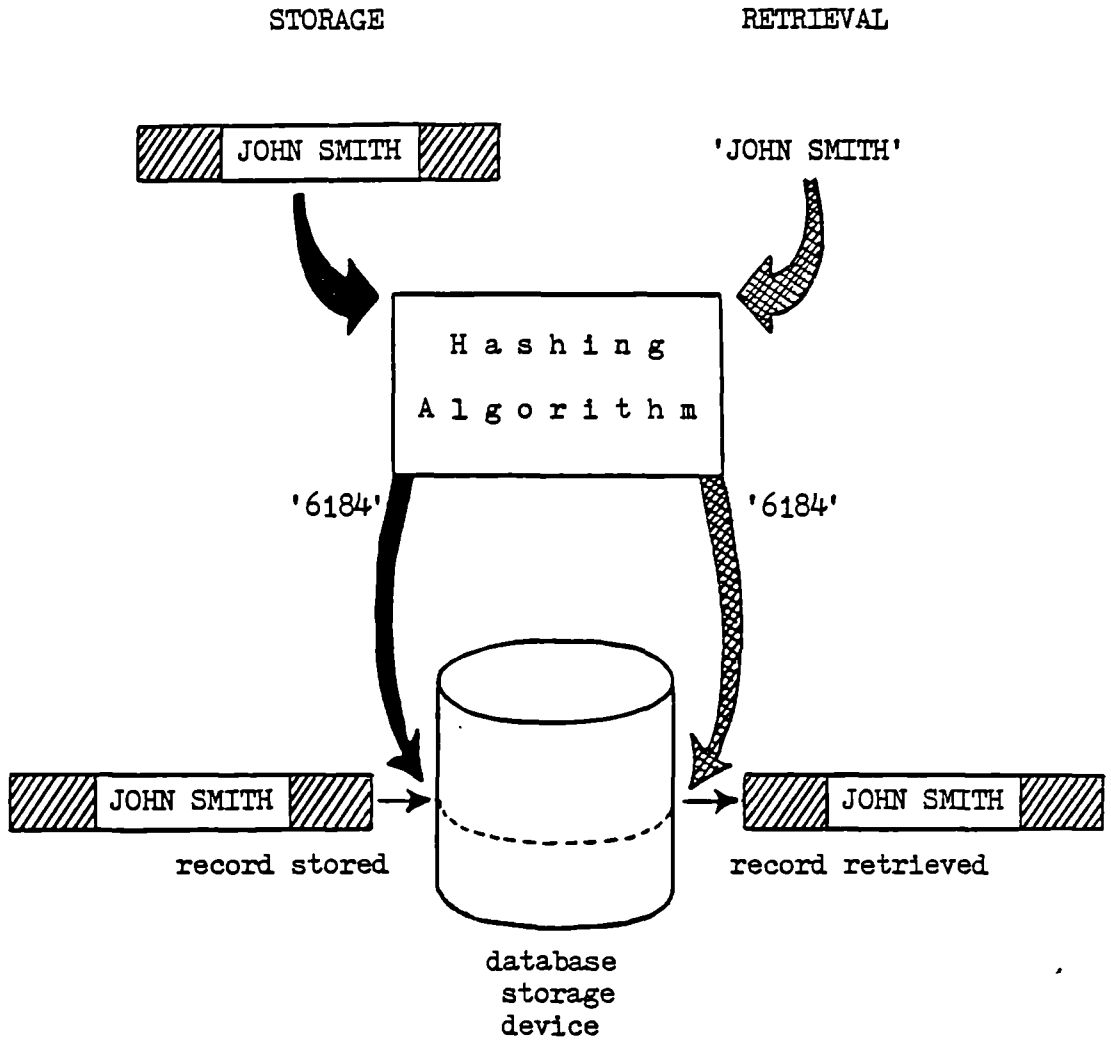


Figure 4.4 The storage and retrieval of records using the CALC facility

Let us now consider how the 'JOHN SMITH' record may be subsequently retrieved. In this case the name 'JOHN SMITH' is presented to IDMS with a request that the corresponding birth record should be obtained. The name is once again submitted to the Hashing Algorithm routine, which, as before, generates the number '6184'. IDMS then uses this number as a retrieval address, and so gains rapid access to the required 'JOHN SMITH' record.

It should be said that in reality the CALC facility is rather more elaborate than the above description may suggest. Thus, for example, IDMS can readily cope with multiple occurrences of the 'JOHN SMITH' record, if this is required. Fortunately for the IDMS user all the technical complexities of the facility are carefully hidden behind the IDMS interface. In his program the user merely needs to write statements such as 'STORE CALC BIRTH-RECORD' and 'OBTAIN CALC BIRTH-RECORD' to invoke the necessary storage and retrieval functions.

4.3.2 The SET Concept

A database will normally be used to hold records of different kinds. For example, a database containing birth records might also typically contain marriage records. Although these records describe isolated events they will nevertheless possess implicit relationships with each other: for example, two particular birth records could relate to a brother and a sister. In order that the database should

accurately model the 'real world' it is therefore necessary for such inter-record relationships to be introduced explicitly into the database. One could, for example, envisage that in a database of birth records the records for each group of siblings might be connected together in some way, e.g. via a chain of address pointers. Additionally, if marriage records were present then it would be possible to connect up each group of sibling records to the marriage record of the parents.

An example of such linkages is shown in Figure 4.5. The central marriage record is shown to be linked to three birth records, suggesting that here there is a family with three children. The unlinked marriage record on the left would suggest a childless marriage and the unlinked birth record a 'missing' marriage (e.g. the marriage may have been recorded elsewhere).

This database organisation can now permit us to carry out some interesting and elaborate operations. For example, as before we can use the CALC facility to locate a particular birth record, e.g. the 'JOHN SMITH' record. However, we can then proceed to use the inter-record links to give us access to the birth records of JOHN SMITH's brothers and sisters and also to the marriage record of his parents.

This very important record-linking capability is provided in IDMS by the SET facility, and it should be noted that the term 'set' when used in this context does not correspond with its use in conventional

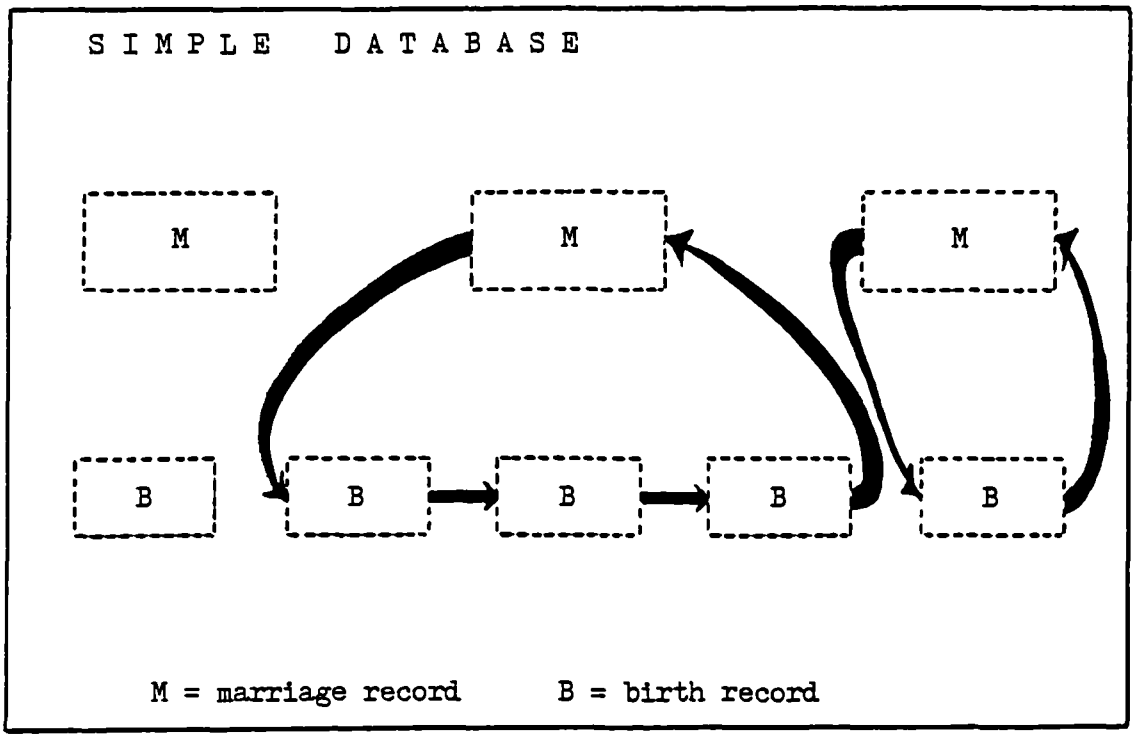
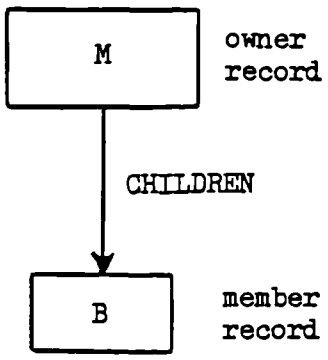


Figure 4.5 The linkage of records in a simple database



1:n relationship

Figure 4.6 The formal representation of an IDMS set

mathematical set theory. Each set in the database is effectively a cluster of records of one type (e.g. birth records) connected to a single record of some other type (e.g. a marriage record). A formal method of describing this relationship is illustrated in Figure 4.6. The 'single' record is referred to as the owner record and the 'cluster' record as the member record. The arrow connecting the two records serves to indicate that a 'set relationship' exists between them, and it is always drawn so as to point from owner to member. This simple method for representing a set enables highly complex database structures to be expressed in the form of simple diagrams. These representations are sometimes referred to as 'Bachman diagrams', after their originator, Charles Bachman (5). It should be emphasized that although such a formal representation shows each set as possessing only a single member record, the actual occurrence of the set in the database can contain any number of member records.

In order to distinguish clearly between formal and actual representations in this thesis the convention has been adopted that in formal representations, e.g. as in Figure 4.6, records are drawn with solid edges; where there is a requirement to illustrate actual sets and records in a database, e.g. as in Figure 4.5, records are drawn with broken edges.

There are some additional, important properties of sets which should be noted:

1. each set type is given a name, which is used when instructing IDMS to execute a set operation. For example, the set shown in Figure 4.6 is called 'CHILDREN', and a typical set instruction would be 'OBTAIN FIRST BIRTH-RECORD WITHIN CHILDREN'.

2. the relationship between owner and member records is strictly 1:n. Thus, in the 'CHILDREN' set a given marriage record can own any number of birth records, but a given birth record can be owned by only one marriage record. In some situations (though not, in fact, here) an n:n relationship is required: the procedure for achieving this is examined in Section 4.3.3.

3. a particular record type may be an owner record in one or more set types, and it may simultaneously be a member record in one or more other set types. This gives the IDMS user a high degree of flexibility, and it enables him to assemble in 'meccano-style' fashion a record and set organisation which uniquely matches his needs. Each set type represents the simplest possible hierarchical structure, i.e. a two-level tree. But by connecting several of these together one can create whatever structure is required: n-level trees, networks, etc.

4.3.3 A Population Database

One of the most technically difficult problems of nominal record linkage is how to structure the linked records. For person-based record linkage this can be a relatively simple task, especially where the linkage involves only the sorting and merging of records from nominal lists (6). But for family-based record linkage the situation is entirely different, and the problem is essentially one of deciding how to organise one's records in a population database. Let us examine the contribution which IDMS can make to the solution of this problem.

There are considered to be two primary requirements of a population database, as follows:

1. it must be capable of modelling real-life genealogical relationships: father, mother, cousin, step-son, etc.

2. it must provide access to subgroups of the people in the database which are of potential interest, e.g. those born in a particular year, those with specific occupations, etc. It should also provide corresponding access to subgroups of the families which are of potential interest, e.g. those for which the marriage was registered in a particular year.

I shall deal with these two requirements in turn.

The main problem in handling genealogical structures is that they consist of networks, rather than simple hierarchies (7). Thus, for example, an individual can be related to a particular ancestor simultaneously via his father and via his mother. Fortunately, it is possible to model these potentially complex structures using a surprisingly simple arrangement of set and record types: this is illustrated in Figure 4.7. In this arrangement the 'CHILD' set is similar to the 'CHILDREN' set already discussed: thus, it connects each individual to his family of origin. The main difference here is that the 'PERSON' and 'FAMILY' records are able to have a broader significance and can contain potentially more information than is present in the corresponding birth and marriage records. For example, a 'PERSON' record could contain the age at death and a 'FAMILY' record the number of offspring.

Two additional sets, 'FATHER' and 'MOTHER', are used to connect each 'FAMILY' record to the 'PERSON' records of the marriage partners. But it should be observed that in these sets 'PERSON' is the owner record and 'FAMILY' is the member record. It is therefore possible for a male 'PERSON' to own more than one 'FAMILY' in the 'FATHER' set, and similarly for a female 'PERSON' in the 'MOTHER' set: this set organisation will therefore cope happily with remarriage. It can also cope readily with illegitimacy. For example, an illegitimate person's family of origin could be represented by a 'FAMILY' record which was not linked to a 'PERSON' record in the 'FATHER' set.

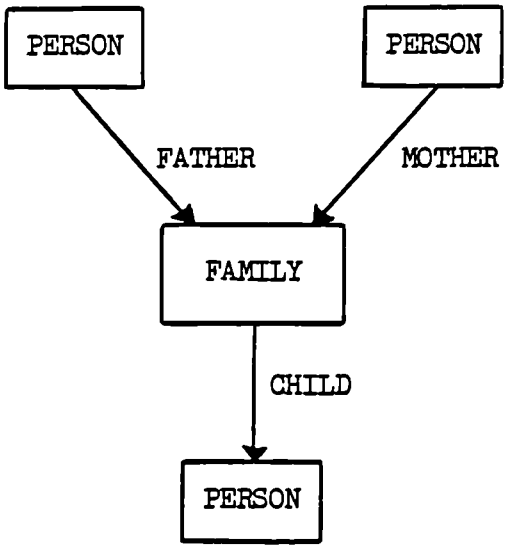


Figure 4.7 A set diagram for family structures

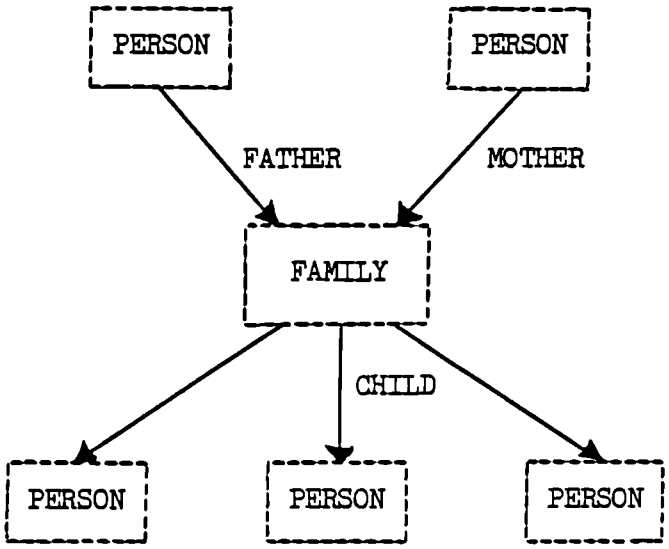


Figure 4.8 Population database: a typical family

Let me now consider some real-life examples of family structures to observe how this arrangement of sets and records works in practice. Figure 4.8 shows how a typical family with three children would be represented in the database. The father and mother are shown as having married only once, i.e. each of their 'PERSON' records owns only one 'FAMILY' record. Figure 4.9 illustrates how the brother-in-law relationship is implicitly represented in the database. The family on the left is shown to have two children, X and X's sister. X's sister is married, and so she owns a 'FAMILY' in the 'MOTHER' set which is simultaneously owned in the 'FATHER' set by her husband, X's brother-in-law.

A more complex example, taken in fact from my own family-tree, is shown in Figure 4.10. The complexity arises here because the male 'PERSON' on the left (my great-great-grandfather) married twice, and at his second marriage married his son's wife's sister! One therefore can observe that he owns <u>two</u> 'FAMILY' records in the 'FATHER' set. The first 'FAMILY' record owns a 'PERSON' record in the 'CHILD' set: this represents his son. The son is married, and so he owns a 'FAMILY' record in the 'FATHER' set which is simultaneously owned in the 'MOTHER' set by his wife. She has a sister, and they are therefore owned by the same 'FAMILY' record in the 'CHILD' set. Finally, the circle is completed by the sister's marriage, in which she owns a 'FAMILY' record in the 'MOTHER' set which also happens to be the second 'FAMILY' record owned in the original 'FATHER' set.

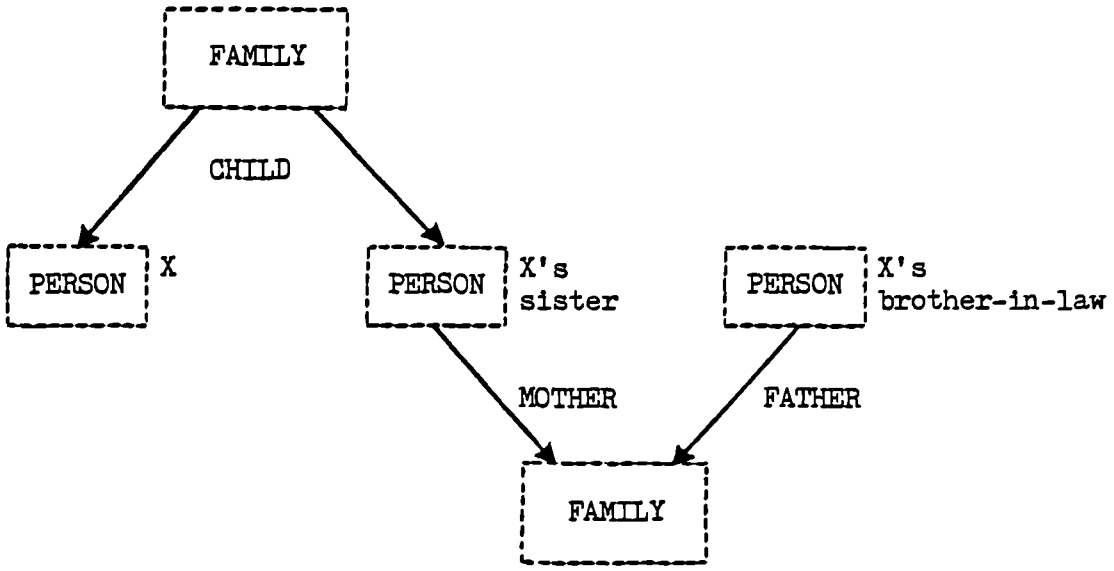


Figure 4.9 Population database: the brother-in-law relationship

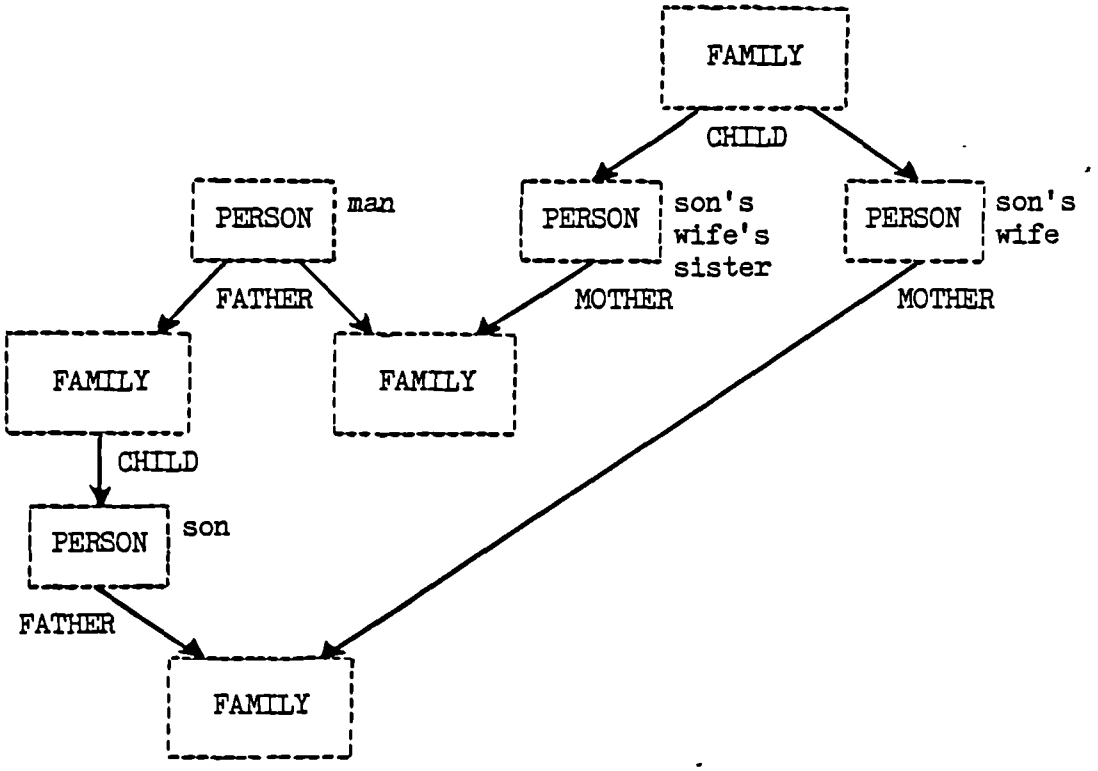


Figure 4.10 Population database: a complex relationship

It should be clear from the above examples that a moderately simple set organisation, employing only two record types and three set types, can be used to model even the most complex genealogical relationships. The central role of the 'FAMILY' record in this arrangement is of particular importance and should be noted. Apart from its more practical functions of sustaining inter-person links and providing a family dossier mechanism, it is also a crucial component in the development of a family-based record linkage strategy. The full implications of this will be explored in later chapters.

It is now necessary to turn to the second of the two requirements of a population database, viz. that it should provide access to subgroups of people and families in the database which are of potential interest. In order to achieve this it is necessary to take the 'PERSON'-'FAMILY' structure shown in Figure 4.7 and extend it with additional record and set types to provide the required access routes. Figure 4.11 illustrates how this might be done.

Consider how such an extended database structure could be used in a particular demographic study. Suppose that one wished to compare the mortality of the people born in two specific years, say 1771 and 1801. To do this it would be necessary to gain access to the 'PERSON' records of all those born in the specified years so that one could then determine their ages at death. The required access route for this is provided by the 'COHORT' record and 'PERSON-COHORT' set shown in Figure 4.11 (8). In the actual database there will be a 'COHORT' record for each year, and, as indicated, each such record is stored

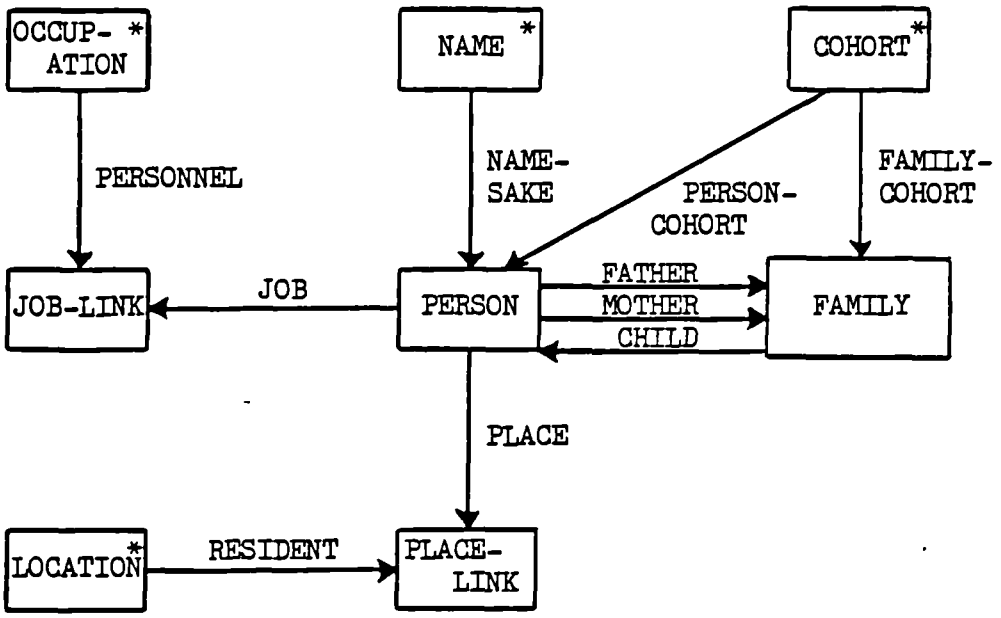


Figure 4.11 An extended set diagram for family structures, showing access routes
(* = CALC record)

and retrieved using the CALC facility. One can therefore readily gain access to any desired 'COHORT' record, say the one for 1771, merely by presenting the appropriate year value to IDMS. Now in the 'PERSON-COHORT' set this record will own the 'PERSON' records for the people born in 1771. It is therefore a simple matter, by repeatedly using a statement such as 'OBTAIN NEXT PERSON WITHIN PERSON-COHORT', to gain access to the required 'PERSON' records. One would then gain access to the 'PERSON' records for those born in 1801 by executing a parallel sequence of operations.

The 'FAMILY-COHORT' set links 'COHORT' and 'FAMILY' records in a similar fashion. In this set the 1771 'COHORT' record would own the 'FAMILY' records for marriages registered in that year. It is therefore possible to use this set to study family characteristics, e.g. the number of offspring, over time.

A simple name index for the population database is provided by the 'NAME' record and the 'NAMESAKE' set. A given 'NAME' record, e.g. the 'JOHN SMITH' record, would own the 'PERSON' records for all people with that name. It would therefore be a relatively simple operation to trace a family-tree by entering the database via a 'NAME' record, locating a specific 'PERSON', and then using the 'FATHER', 'MOTHER' and 'CHILD' sets to search for relatives of the initial named individual. But it should be stressed that in practice a rather more elaborate name indexing facility is required. In particular, it needs to be able to cope with name variations and with a woman's change of name at marriage. Such a name indexing facility is described fully in

Chapter 9. The simple index illustrated in Figure 4.11 operates on the assumption that each person is always referred to in precisely the same way.

Two additional access routes are shown in Figure 4.11: these are concerned with occupations and locations. I shall consider first the provision of a simple access route for occupations. The requirement here is to be able to select a particular occupation, e.g. farmer, and locate all the 'PERSON' records for people with that occupation. However, the problem is rather more complicated than it may initially appear since each person in the database may have had several occupations during their lifetime. It is therefore not possible merely to make the 'PERSON' record a member in a 'PERSON-OCCUPATION' set. (Compare this with the PERSON-COHORT set: here the simple solution is permissible since no person in the database can have been born in more than one year.) This complex relationship between occupations and people is, in fact, an n:n relationship. In the set terminology one would say that an 'OCCUPATION' record should be able to own any number of 'PERSON' records and a 'PERSON' record any number of 'OCCUPATION' records. But such an n:n relationship is not permitted in an IDMS set: one record must be the owner and the other the member. The solution to this problem is to introduce a new record type, 'JOB-LINK', which can exist as a member of a set ('PERSONNEL') in which 'OCCUPATION' is the owner, and which can simultaneously exist as a member of another set ('JOB') in which 'PERSON' is the owner. This arrangement now permits each 'PERSON' in the database to own a 'JOB-LINK' record for each of his occupations. Each 'JOB-LINK' record

will be linked to its appropriate owning 'OCCUPATION' record, and so the access route is complete.

I now turn finally to the provision of an access route for locations. Here the requirement is to be able to select a particular location, perhaps a village, and locate all the 'PERSON' records for people who lived there. Since the relationship between locations and people is also n:n, i.e. people live in different places at different times, one must provide an appropriate access route, as for occupations. This is also shown in Figure 4.11.

I have now completed my examination of the way in which IDMS can assist the organisation of records in a population database. I shall return to this topic in Section 9.2, when I shall consider the design of a fully working population database. Later, in Section 11.4, I shall examine proposals for using a dual arrangement of IDMS and a relational database system to provide more generalised access to the contents of the database.

4.3.4 A Glossary of IDMS Terms

In the following chapters it will be necessary to make use of certain IDMS/CODASYL technical terms. Instead of defining these in a piecemeal fashion as they occur, it should prove more helpful to have the definitions presented in advance in the form of a glossary: this strategy should also assist the reader should he subsequently wish to

review or reference the definitions. However, it must be stressed that the glossary which follows is not intended to be an exhaustive and definitive list of IDMS terms. Rather, its prime function is to provide a suitable technical preamble for the later chapters. The definitions which follow are also included in the more comprehensive glossary provided in Appendix A.

AREA — The total physical storage space used to hold a database can be logically subdivided into a number of parts, called areas. Broadly speaking, a given area will contain those records and sets which constitute some major subdivision of the database. In the present system design the database subdivides into three areas: the first containing the name directories, the second the source records and the third the population structures.

CALC — The CALC facility enables efficient access routes into the database to be established. The method of storing and retrieving each CALC-type record is commonly based on the 'hashing' of some identification field within the record. (For a fuller explanation see Section 4.3.1.)

DDL — The DDL, or Data Description Language, is the language with which the designer of a particular database initially communicates the structure of the database to the database system. For IDMS the designer must provide details of the

areas, records and sets which make up the database. The description of the whole database is contained in the Schema (q.v.), and the Schema DDL is the appropriate language for preparing this. The description of some part of the database which is required by one or more application programs is contained in a Subschema (q.v.), and the Subschema DDL is the corresponding language for preparing this.

DMCL — The DMCL, or Device Media Control Language, is the language with which the designer of a particular database communicates the physical storage and input/output buffering arrangements for the database. While the DDL (q.v.) is strictly concerned with the logical organisation of the database, the DMCL is concerned with its physical organisation.

DML — The DML, or Data Manipulation Language, is the language with which the writer of a database application program expresses the operations which he wishes to have carried out on the database. To a COBOL programmer, for example, the DML effectively gives him an enlarged programming vocabulary with which to write his programs. Typical DML database access statements available to the programmer are FIND, STORE, CONNECT and ERASE, and each of these when executed will operate on a particular record in the database.

DIRECTORY Detailed information defining the structure and properties of the database has to be communicated to the database system using the special DDL and DMCL languages (q.v.). In order to make use of this information the database system creates its own private database, called the Directory, in which it stores the relevant information. This Directory is subsequently referred to when, for example, IDMS is required to process DML statements during the compilation of a program.

RECORD Within a database the fundamental unit of data storage and retrieval is the record. Each type of record has its own identifying name, which the user employs in his communications with the database system. A record can have a substructure, i.e. it can be made up of several smaller data items, each with its own identifying name. For example, a PERSON record could contain a NAME data item, a SEX data item, a DATE-OF-BIRTH data item, and so on.

SCHEMA A schema is a description of a whole database, expressed in terms of the areas, records and sets of which it is constituted. It is effectively a master-plan, describing in full the properties of each record type and the potential set relationships which can exist between them. A database designer communicates the schema to the database system using the Schema DDL language.

SET Within a database the individual records can be associated with each other by means of a set relationship. A particular set type, as defined in the schema (q.v.), will most usually define one particular type of record as the owner record and some other type of record as the member record. A given occurrence of this set in the database will then typically consist of one record of the owner type connected to several records of the member type. Each set type has its own identifying name, which the user employs in his communications with the database system. (For a fuller explanation see Section 4.3.2.)

SUBSCHEMA Individual application programs may not need to be aware of the complete database structure, and so it can be helpful to define a number of simplified 'views', each of which contains only those areas, records and sets needed by the individual programs. Each such simplified 'view' is called a subschema, and the database designer communicates a description of this to the database system using the Subschema DDL language.

NOTES

1. Most of the writers of present-day programming language teaching texts appear to regard it as essential that the concepts of structured programming should be taught, and not merely the individual elements of the language. Indeed, the titles of these texts increasingly reflect the changing emphasis, being often of the form 'A guide to structured programming in language X', or something of this kind. A typical example, and a book which can be strongly recommended, is Daniel McCracken's 'A simplified guide to structured COBOL programming' (McCracken 1976).

2. There are methods by which IMS can be used to handle the more complex network data. However, this is achieved in a somewhat cumbersome way, and one which is alien to the essentially hierarchical philosophy of IMS.

3. It should be noted that the use of the precise relational terminology is being avoided here, since it is somewhat obscure and could confuse. For example, the familiar two-dimensional table is referred to as a 'relation', and a record is called a 'tuple' (Date 1986, 96-100).

4. CODASYL is an abbreviation for Committee on Data Systems Languages. For an introduction to the main CODASYL concepts, presented 'in tutorial fashion', see Taylor and Frank 1976. A more exhaustive description is provided in Olle 1980.

5. Bachman 1969, cited in Olle 1980, 24.

6. Other kinds of person-based record linkage which are equally simple to organise include 'track editing' (Herlihy 1981, 134) and the 'dragnet method' (Roy 1982). In each case the objective is to search through the total, available information and abstract for linkage only those items which relate to specific 'targets' in which one is interested. There is, therefore, no attempt to link <u>all</u> the available information to produce a composite, 'real world' view.

7. It is because of this network property that people tracing their family trees, and also those doing manual family reconstitution, can have significant problems organising their records. Attempts to use conventional, hierarchical filing systems for this are fraught with difficulties.

8. In the context of demographic analysis the term 'cohort' refers to a group of individuals who have experienced the same event, e.g. their birth, within the same time period.

CHAPTER 5 IMPLEMENTATION OF THE RECORD LINKAGE SYSTEM: INTRODUCTION

In the last chapter I examined in a general way how one should cope with the complexity problems which can arise in program development. I also looked in detail at the way in which a database management system can be used to model 'real world' structures, and, in particular, how it can assist the organisation of a population database.

It is now necessary to confront some of the more practical problems which arise and the choices which must be made when embarking on the implementation of an operational record linkage system. Central to my overall development philosophy is the belief that it is not acceptable merely to have produced a sound system design. Of equal importance is the creation of a practicable scheme for translating the design into a workable system. This crucial task demands both the selection of appropriate implementation 'tools' (e.g. programming languages, database management systems, etc.) and also the skills needed to exploit the tools which are selected. In this chapter I shall therefore consider the issues which arise in the selection of a suitable implementation environment and the establishment of appropriate standard programming and database methods. Given that family-based record linkage is a computing problem of such complexity, the importance of this task cannot be over-estimated.

5.1 SELECTING THE IMPLEMENTATION ENVIRONMENT

5.1.1 Computer Sites and Systems

At an early stage in a computer project one has to make a number of crucial decisions about which machines and implementation tools one is going to use. For example, if one is to use a programming language which one is likely to be the most suitable for the task in hand: COBOL, ALGOL, FORTRAN, PASCAL, BASIC, or perhaps some other alternative? Such a choice will normally, in practice, be limited by the range of compilers which is provided on the local, available machine. However, where one has special requirements which cannot be satisfied locally one may decide to make use of remote facilities, despite the additional difficulties which may be incurred in their operation.

In addition to assessing the suitability of competing, alternative systems for the task in hand one may also need to consider how universally available such systems are, particularly where one's aim is to develop programs which can be readily used elsewhere. By writing one's programs in COBOL or FORTRAN, for example, one can expect to be able to compile and run them, perhaps after minor modification, on most available machines (1).

The selection of a suitable implementation environment must therefore be made with due care, and in the next two sections a brief

account is given of the ways in which for the current research, which was being carried out in Edinburgh, the various options presented themselves and were assessed.

5.1.2 The Database Management System

The most important decision which had to be made was the selection of a suitable database management system. In view of the need to be able to represent complex network structures in a population database it was clear that a CODASYL-type system, such as IDMS, would be able to provide the required modelling facilities. In addition, for record linkage work it was felt that a system which viewed the 'record' as the fundamental unit of data manipulation would be especially desirable. This, again, favoured the choice of a CODASYL-type system.

At the time when the decision about database systems was being made (1975-6) there was no suitable relational database system readily available, and so this option did not present itself. However, even if such a system had been available there is significant doubt about its suitability for a record linkage application, for the reasons given in Section 4.2.3. In particular, since the fundamental unit of data manipulation for a relational system is the 'table', rather than the individual record, it is probable that the consequent mismatch between the relational view and the record linkage program view would have caused difficulties (2).

In the event it was decided that a CODASYL-type system would most adequately satisfy the requirements, and IDMS was therefore chosen (Welford 1977B). During the first year of the implementation (Nov. 1976 - Nov. 1977) it was necessary to use IDMS remotely at Newcastle on the Northumbrian Universities Multiple Access Computer (NUMAC). This service was based on IBM 360/65 and 370/168 machines. But subsequently the Edinburgh Regional Computing Centre (ERCC) provided an IDMS service on the local ICL 2980 machine running under the VME/B operating system (3).

5.1.3 The Implementation Language

Once a particular database management system has been selected the choice of which implementation language to use is then restricted to those languages which have been tailored to interface with it. In the case of IDMS at Edinburgh the choice was between ICL 2900 COBOL (4) and Edinburgh FORTRAN (5).

It was decided to use COBOL rather than FORTRAN as the implementation language for several reasons. In the first place, the COBOL language is oriented towards commercial, record-handling applications, and, as such, it provides comprehensive facilities for declaring complex record structures. By contrast, FORTRAN is oriented towards scientific applications, and its facilities for organising records are primitive. In the second place, the available version of COBOL has language facilities which assist the writing of structured

programs (6), while Edinburgh FORTRAN is lacking in such facilities. Thirdly, COBOL has a number of useful high-level facilities, such as automatic table-searching, which in a FORTRAN program would need to be implemented by the programmer. Finally and not insignificantly, the various CODASYL data description languages which are used by the programmer to communicate with IDMS have a syntax which is closely modelled on that of COBOL. It is possible, therefore, for a programmer to maintain a more unified and consistent view of the database if he is also writing his programs in COBOL.

5.2 PROGRAMMING METHODS

In Section 4.1.3 I examined a number of strategies for simplifying the organisation of programs. I identified, in particular, the valuable contribution which can be made by employing a 'top-down' design approach and also the use, at a more detailed level, of structured programming techniques. I shall now consider the application of these methods within the present implementation environment, and with reference, in particular, to my use of the COBOL programming language.

5.2.1 Systems, Subsystems and Modules

The main characteristic of the 'top-down' design approach is the progressive disaggregation of a complex problem into simpler problems, and ultimately the specification of a large number of relatively simple routines which can be coded. In view of the potential complexity of the overall task of disaggregation it can be helpful if the process is formalised in some way. A convenient method of doing this is to define a number of discrete levels of disaggregation, and in the present design I make use of three main descriptive terms, viz. 'system', 'subsystem' and 'module'. I shall now clarify the meanings which I am attaching to these terms.

The term 'system' is the most global, and it encompasses all the program code which has been developed within the present research.

The system, as defined, consists of a number of independent programs, called 'subsystems'. Each subsystem operates in isolation from the others (7), and it has responsibility for some integral part of the total record linkage operation. There are six subsystems in all, and, for example, the Directory Prime Subsystem has responsibility for the setting up and organisation of the Christian name, job-name and other name directories. Such subsystems are somewhat analogous to the main functional units of a car engine, such as the petrol pump and the carburettor: thus, they have a limited but integral part to play in the operation of the whole system.

The program code within each subsystem is further disaggregated into a number of COBOL program 'modules', each of which can be compiled separately and which at execution time is invoked by means of a COBOL 'CALL' statement. Within the total system there are 110 such modules.

The disaggregation of each subsystem into individual modules is organised in such a way that each module has a limited area of responsibility and a limited number of functions to carry out. For example, the Source Translation Subsystem, which provides initial handling of the nominal records, contains 26 modules, of which the following are a selection:

SACEBPROCESS - which controls the translation of a block of census household records.

SBCEHPROCESS - which controls the translation of one census household record.

SCCEPPROCESS - which controls the translation of one census household occupant record.

N3STPROCESS - which handles the translation of people's names in both census and parish register records.

T1STPROCESS - which is responsible for handling all date and age information.

DWDBSEARCH - which has responsibility for communicating with IDMS in order to search the Christian Name, Surname and other name directories in

the database. As such, it has the distinctive property of being the only module in the subsystem to interface directly with IDMS.

The policy adopted for naming such modules is described in Note 8.

5.2.2 Organising Inter-Module Interfaces

One of the penalties which must be paid for deciding to disaggregate a system into a number of subsystems, and thence into a larger number of modules, is the problem of ensuring that the separate parts can ultimately be brought together and made to function as an integrated system. It is fairly clear, for example, that a system made up of 110 modules could have severe inter-module communication problems, both with the transferring of control and with the transmission of information. It is essential, therefore, that a clear and disciplined strategy for organising inter-module communication should be firmly established if confusion and system failure are to be avoided.

In order to illustrate the particular strategy which has been adopted I shall examine just one inter-module interface, that between the modules 'SCCEPPROCESS' and 'N3STPROCESS', which were referred to briefly above. Essentially I shall be concerned with the process by which module 'SC' (i.e. 'SCCEPPROCESS') invokes 'N3' (i.e. 'N3STPROCESS') to carry out a particular task. The overall function

of the 'SC' module is to supervise the source translation processing of one census household occupant record, a record such as:

C6/P/JOHN WILSON/HEAD/MAR/49//FARMER/DURHAM,STOTFOLD

The structure of this record is precisely as it appears in the source document, except that it has been prefixed with two record identification tags. The first, 'C6', identifies the record as a census record for 1861, and the second, 'P', identifies it as a person record. The translation of this record involves the scanning of the various fields and their conversion into appropriate code values: a detailed description of this process will be provided in Chapter 8.

The first action of module 'SC' is to check the initial record identification tags to ensure that they are correct. Assuming that they are it then proceeds to invoke other modules to check and translate the other fields in the record. The first module to be called is 'N3', and this will be responsible for handling the name field, 'JOHN WILSON'. The COBOL statement which appears in 'SC' and which causes 'N3' to be invoked is as follows:

```
CALL 'N3STPROCESS'  USING  NAME-BLOCK
                           INPUT-BLOCK.
```

In this statement the string 'N3STPROCESS' identifies the module to which control is to be transferred, and the parameters to the call are identified as 'NAME-BLOCK' and 'INPUT-BLOCK'. In COBOL there is no concept of 'global' data, i.e. data which may be accessed directly

from more than one module. The only convenient method for communicating information from one module to another is therefore via the parameters in the 'CALL' statement.

Let us consider first what information is passed between modules 'SC' and 'N3' via the 'NAME-BLOCK' parameter. A declaration for 'NAME-BLOCK' appears in the 'WORKING-STORAGE SECTION' (9) of module 'SC'. The structure is as follows:

```
01  NAME-BLOCK.
    02  NB-FUNCTION              PIC X(2).
        88  NB-CENSUS-INIT       VALUE 'CI'.
        88  NB-CENSUS-NAME       VALUE 'CN'.
        88  NB-BAPTISM-INIT      VALUE 'BI'.
        88  NB-BAP-CHILD-FORENAMES
                                  VALUE 'BC'.
        88  NB-BAP-FATHER-FORENAMES
                                  VALUE 'BF'.
        88  NB-BAP-MOTHER-FORENAMES
                                  VALUE 'BM'.
        88  NB-BAPTISM-SURNAME   VALUE 'BS'.
        88  NB-MARRIAGE-INIT     VALUE 'MI'.
        88  NB-MARRIAGE-NAME     VALUE 'MN'.
        88  NB-MAR-FATHER-NAME   VALUE 'MF'.
    02  NB-NAME-IDENT.
        03  NB-NAME-NUMS.
            04  NB-FNAME-NUM     PIC 9(8).
            04  NB-SNAME-NUM     PIC 9(8).
        03  NB-INITIALS          PIC X(2).
        03  NB-SEX-IND           PIC X.
    02  NB-SUCCESS-IND           PIC X.
        88  NB-NAME-FOUND        VALUE 'Y'.
        88  NB-NAME-NOT-FOUND    VALUE 'N'.
```

This data storage space is used by 'SC' to indicate to 'N3' which operation is required. It is subsequently used by 'N3' to communicate the results of the operation back to 'SC'. The three major components of the parameter are as follows:

1. NB-FUNCTION. This is the variable which is used by a caller to indicate to 'N3' which operation is required. The COBOL term 'PIC X(2)' specifies that this variable may hold any 2-character value, e.g. 'A3'. (10) The range of permissible values for NB-FUNCTION is indicated in the 'type 88' statements which follow. For example, if module 'SC' wishes to indicate to 'N3' that it is to process a name in a census record it must place the code value 'CN' into NB-FUNCTION just prior to executing the 'CALL' statement.

2. NB-NAME-IDENT. This variable is used by 'N3' to return to the calling module details of the name which it has located. The variable has four sub-components, the first two of which are specified as being of type 'PIC 9(8)'. These two variables can therefore each hold an 8-numeric digit value, e.g. '00026559'. (11) The variable 'NB-FNAME-NUM', in fact, contains the code value for the first forename in the name, while 'NB-SNAME-NUM' contains the code value for the surname. The remaining two variables should be fairly self-explanatory. 'NB-INITIALS' contains the initial letters of the first two forenames in the name: for the name 'JOHN WILSON', which has only one forename, the value 'J ' would be returned. The variable 'NB-SEX-IND' contains just one character, which would normally be either 'M' (male) or 'F' (female). Where the gender cannot be unambiguously derived from the forename(s), as is the case with the name 'LESLIE BROWN', then the variable would contain the value 'N' (i.e. null).

3. NB-SUCCESS-IND. This variable is used by 'N3' to indicate the success, or otherwise, of the operation. If it contains the value 'Y' (yes) then this will mean that the processing has been completely successful; if it contains the value 'N' (no) then it will mean that some error has occurred, such as the location in the source record of a surname which could not be found in the directory of surnames.

The second parameter in the 'CALL' statement is 'INPUT-BLOCK'. This is a rather more complex data structure, consisting of storage buffers and control variables for organising the scanning of all types of input. For the purposes of the present illustration, however, I shall consider only two components of the parameter. These are as follows:

```
03  RB-CARD-IMAGE        PIC X(80).
03  RB-REC-PTR           PIC S9(2) COMP-3.
```

These two variables are used by 'SC' to pass a census household occupant record across to module 'N3' for scanning. The record itself is stored in 'RB-CARD-IMAGE', while 'RB-REC-PTR' is a scanning pointer, indicating which character position in the record the processing has so far reached (12). When 'N3' is entered 'RB-REC-PTR' will have the value 6, i.e. it will be pointing at the first character of the 'JOHN WILSON' field, as follows:

```
C6/P/JOHN WILSON/HEAD/MAR/49//FARMER/DURHAM,STOTFOLD
```

'N3' will proceed to scan the record, starting from the position which is indicated, and when it returns control to 'SC' it will leave 'RB-REC-PTR' pointing at the next character which is to be processed, thus:

C6/P/JOHN WILSON/HEAD/MAR/49//FARMER/DURHAM,STOTFOLD

'SC' will then, after unloading the required return values from 'NAME-BLOCK', invoke the scanning routine which is to process the next field in the record, i.e. the relationship field. The method of making this call will precisely parallel that adopted for the previous scanning routine. It will be as follows:

```
CALL 'R2STPROCESS' USING RELATION-BLOCK
                         INPUT-BLOCK.
```

The processing will therefore proceed in a very orderly fashion, with each module having a clearly defined task or set of tasks to carry out, and with the method of passing control information and returning results being made deliberately coherent and simple.

In the above illustration I have considered in detail just one inter-module interface, that between modules 'SC' and 'N3'. In the total system there are 110 modules and 493 such interfaces. The need for the adoption of such a disciplined strategy towards the organisation of the interfaces will therefore be self-evident.

5.2.3 Structured Programming

The main aim of structured programming is to so organise the sequencing of program statements within a module that control always passes in a highly disciplined way from one section of code to another. Ideally the program structure should precisely match the logic of the problem that is being handled: if this is the case then when the programmer needs to refer to his program code he is immediately able to see its overall structure and function. Where such discipline is not present it becomes difficult for the programmer to avoid introducing logical faults into the code, and the result can be an unreliable and unpredictable total system.

Some programming languages lend themselves more readily to structured programming than others (13). In this respect COBOL is moderately good, although it does have limitations.

In order to observe how the concepts of structured programming have been exploited within the present research we will look briefly at a section of the code within module 'N3': this is the part which deals with the first forename in a name. The code is as follows:

```
PROCESS-FIRST-NAME.

    PERFORM CHECK-FOR-DITTO.

    IF      DTT-DITTO-LOCATED

            IF  DTT-DETAILS-AVAILABLE
                MOVE PNB-FNAME-NUM TO NB-FNAME-NUM
                MOVE PNB-FIRST-INITIAL TO IB-FIRST-INITIAL
                MOVE PNB-SEX-IND TO NB-SEX-IND
                                    SEXB-SEX-IND
                PERFORM OPTIONAL-REGISTER-SEX
            ELSE
                NEXT SENTENCE

    ELSE IF SB-STRING-LENGTH = +1

            MOVE SB-STRING TO CHAR-BUF
            PERFORM CHECK-ALPHABETIC
            MOVE CHAR-BUF TO IB-FIRST-INITIAL
            MOVE FIRST-INITIAL-UNWELCOME TO EB-VALUE
            PERFORM PROCESS-WARNING

    ELSE

            PERFORM SEARCH-CN-SN-DIRECTORIES

            IF  DSC-SEARCH-SUCCESSFUL
                MOVE DSC-UNIQUE-CN-SN-NUM TO NB-FNAME-NUM
                MOVE DSC-SEX-IND TO NB-SEX-IND
                                    SEXB-SEX-IND
                PERFORM OPTIONAL-REGISTER-SEX
                MOVE DSC-NORM-INIT TO IB-FIRST-INITIAL.
```

In COBOL terminology this section of code is called a 'paragraph', and it may be executed from a point within the module by issuing the statement:

```
PERFORM PROCESS-FIRST-NAME.
```

Other paragraphs are invoked from within this paragraph by similar 'PERFORM' statements. When the code within a paragraph has been executed control then returns to the statement which follows the

'PERFORM' which initiated the execution.

The function of the 'PROCESS-FIRST-NAME' paragraph is to process the first forename in a name, e.g. the string 'JOHN' in the census household occupant record which was considered earlier. However, it should be noted that in a particular record this field might contain, not a name, but instead a ditto sign (e.g. " or DO), indicating that the first forename is to be assumed to be the same as the one given in an earlier record. Finally, the name field may just have an initial character in it, rather than a full name. Essentially, therefore, the paragraph needs to contain logic to deal with each of these three alternatives. The logic would typically be as follows:

1. If the name field contains a ditto sign then it is necessary to check whether there has been a previous name, and if so, to use the various code values associated with it. If there has not been a previous name then a warning message should be issued.

2. If the name field contains a single character then this should be checked to ensure that it is alphabetic. A warning message should then be issued indicating that without a complete forename subsequent record linkage will be difficult, if not impossible.

3. If the name field contains a character string which is not a ditto sign and which is longer than one character then it

should be searched for in the Christian Name and Surname Directories. If the search is successful the appropriate code values obtained from the directory should be used; if the search is unsuccessful a warning message should be issued.

An inspection of the code in the 'PROCESS-FIRST-NAME' paragraph will reveal that its program structure does accurately reflect the problem logic as just described. The major subdivision of the paragraph into the three distinct logical components is clearly signalled by the 'IF/ELSE IF/ELSE' construction in which the bulk of the code is embedded. Once the major 3-way selection has been made the appropriate 'nested' section of code will then be executed. In the case where a ditto sign has been located, for example, this nested section of code consists of an 'IF/ELSE' 2-way selection, to handle the situations where there are, and are not, previous name details available (14).

There are two minor discrepancies between the problem logic structure, as described, and the 'PROCESS-FIRST-NAME' code. Both concern the issuing of warning messages: the first where there is a ditto sign, but no previous name details available, and the second where a name cannot be located in the Christian Name and Surname Directories. Since these two conditions can arise in many similar circumstances, it would be inconvenient to have to duplicate the coding for the warnings in 'PROCESS-FIRST-NAME' and elsewhere. It is preferable, therefore, to issue each warning centrally, at the point where the condition is originally detected. For example, in the case

where a name cannot be located in the name directories the warning is issued by the search routine itself.

The adoption of such structured programming techniques for the coding of the system has been extremely beneficial. In particular, it has enabled error-free programs to be developed quickly, and the code produced is easy to read and update.

5.2.4 The COBOL COPY Facility

A criticism which is frequently levelled against the COBOL language is that its syntax is unduly verbose. For example, in most other languages to initialise a variable 'X' to zero it is necessary merely to write a statement of the form:

```
X = 0
```

The corresponding statement in COBOL is:

```
MOVE +0 TO X
```

While this example does indicate that there is substance in the criticism it should in fairness be added that COBOL does make up for this by providing a number of facilities which significantly reduce the amount of verbosity which can otherwise be involved in the writing of programs. An example of these facilities, and one which is widely

used, is the COPY facility.

This facility provides a means for copying commonly used sections of data or program into several modules (15). Consider, for example, the declaration of the 'NAME-BLOCK' parameter, which was illustrated earlier in the chapter. This consists of 24 lines of code, and such a declaration must be included in the WORKING-STORAGE SECTION of each module that wishes to call module 'N3', and it must also be included in the LINKAGE SECTION of 'N3' itself. In all, six declarations of the 'NAME-BLOCK' parameter are required within the Source Translation Subsystem.

To use the COPY facility for this declaration it is necessary merely to prepare the declaration once, without the initial '01 NAME-BLOCK' statement, and store it, along with others, as members of a partitioned data file (16), a file which one nominates as the 'COPY Library'. If we suppose that the name of the member containing the 'NAME-BLOCK' statements is 'NAMBLOCK' then in order to have these statements inserted in any module it is necessary merely to include the following statement at the appropriate point:

```
01 NAME-BLOCK. COPY NAMBLOCK.
```

When the COBOL compiler encounters this statement it accesses the file which has been nominated as the COPY Library and selects the member called 'NAMBLOCK'. It then proceeds to copy into the module the appropriate declaration statements from 'NAMBLOCK'.

The COPY facility can clearly reduce the amount of code which needs to be written, and, as a bonus, it automatically guarantees that each module uses the correct copy of a given parameter declaration. The facility has been used extensively in the development of the linkage system, the COPY Library eventually having as many as 189 members.

5.3 DATABASE METHODS

In Section 5.2 I explored the application of various design and programming methods within the context of my use of the COBOL programming language. What I shall now proceed to do is to carry out a parallel analysis with regard to the use of database methods and within the context of my use of the IDMS database management system. Unfortunately, since the use of CODASYL-type databases was still relatively in its infancy during this development, there were no firmly established guidelines as to how a database and the program's interface to the database should be best organised. What follows, therefore, is an account of the particular strategies and approaches which I developed for the present record linkage application.

A general programming principle, which was propounded in Section 5.2.1, recommends that, as far as possible, each subsystem should have only a limited and clearly circumscribed part of the total problem to deal with, and similarly, but even more so, for each module of code.

A similar principle is seen also to apply with respect to the organisation and use of the database. It can be recommended therefore that, as far as possible:

1. a given subsystem should need to be aware only of that part of the database which is essential to its operations.

2. within a given subsystem as few as possible of the modules should have a direct interface with the database system. As already mentioned, within the Source Translation Subsystem only one of the 26 modules has a direct interface with IDMS.

3. for those modules within a subsystem which have a direct interface with the database system each one should need to be aware of as limited a part of the database as possible.

Certain features of IDMS are particularly helpful in enabling the design objectives enshrined in these recommendations to be achieved. In the following sections I shall therefore examine the nature of these features and my overall strategy for using IDMS.

5.3.1 The Subdivision of the Database into Areas

IDMS provides facilities which enable the storage space used to hold a database to be subdivided logically into a number of parts, called 'areas'. While records and sets are intended to model 'real

world' entities and relationships, respectively, areas are not normally expected to have such 'real world' significance. Instead, they are provided chiefly to assist the efficient organisation of the storage space used for the database.

However, for the purposes of the present research the concept of an area was a useful one for helping to reflect a particular 'real world' view within the database. The total database, as designed, consists of three quite independent 'databases', having no physical connections between them. The first contains the name directories, the second the unlinked source records, and the third the population database. It therefore seemed natural to allocate each of these 'databases' to its own area, and this arrangement has proved to be a worthwhile one. IDMS has facilities which enable it, for example, to permit a given program to write to a particular area, but only to read from another. This property of an area fits in well with the way in which the individual subsystems need to view the separate databases. Thus, for example, the Record Linkage Subsystem clearly needs to be able to write information to the population database area, but it only needs to be able to read information from the areas containing the name directories and the unlinked source records. Other subsystems have their own, differing access requirements.

5.3.2 The Use of Several Subschemas

While the concept of an 'area' can provide a potentially useful means of disaggregating a database organisation which may be complicated into a number of simpler parts, a corresponding simplification can be achieved by the use of another IDMS feature, viz. the concept of the 'schema' and its potential reduction into a number of 'subschemas'.

A schema is essentially a description of a whole database, expressed in terms of the areas, records and sets of which it is constituted. In the case of the present record linkage database the schema contains 3 areas, 49 record types and 46 set types.

A particular subsystem will not, in general, need to have access to all the items described in the schema, and so IDMS provides a facility which enables a subsystem to have a simplified 'view' of the database, containing only those items which are actually needed. Such a simplified view is called a 'subschema'. Since different subsystems may each need their own individual views, based on their functional requirements, one can create an equivalent number of subschemas, each individually tailored to those requirements. A useful analogy may be to think of an Ordnance Survey map of Great Britain as constituting a type of schema. From this Ordnance Survey map one can derive a number of special-purpose maps, by choosing to use only part of the total information provided. For example, one could create a road map, a rail map, a hikers' map for the Lake District, and so on. Each such

map would correspond with a subschema: it would be geared to a particular type of use, and would not need to contain information which was extraneous to that use.

A significant advantage of using a range of subschemas is that they make operations simpler for the individual programs that use them. In a similar fashion a special-purpose map can be much more convenient to use than a fully detailed one.

A particular benefit that the use of subschemas brings is a measure of 'data independence'. The concept of data independence relates to the ability to modify the structure of a database (effectively the schema), without the need for widespread changes to the programs which access the database. Where there is little data independence, for example, it may prove necessary to recompile every module in the system each time a change is made. Fortunately, the use of subschemas can greatly reduce the amount of program change and recompilation which is necessary. Should, for example, a particular modification to the schema affect only one of the subschemas, then only the modules which use that subschema may need to be changed and recompiled.

For the present record linkage system each of the five major subsystems (17) was provided with its own subschema. The subschema for the Source Translation Subsystem, for example, has access only to 1 area, 10 record types and 5 set types; the subschema for the Record Linkage Subsystem, by contrast, has access to all 3 areas, 39 record

types and 38 set types.

5.3.3 The Minimisation of a Subsystem's Interface to IDMS

I shall now begin to explore the actual method of invoking the IDMS system from within each subsystem. The main objective will be to identify strategies which are consistent with my overriding 'top-down' design orientation.

Within each subsystem an attempt has been made to restrict the access to IDMS to as few modules as possible. This is good programming practice, in that it minimises the amount of disruption which is caused when modifications to the structure or use of the database are required. It can additionally facilitate the complete transfer of the system to an alternative database management system, should such an extreme action be required.

Let us observe how this policy has been put into effect in the case of the Source Translation Subsystem. Only one module, 'DWDBSEARCH', has access to IDMS, and each of the other 25 modules in the subsystem can access the database only indirectly, by calling this module. In Section 5.2.2 I examined the processing of census household occupant records, and considered in detail how a name field, 'JOHN WILSON', would be handled by the module 'N3STPROCESS'. Although 'N3' has responsibility for unpacking and checking the various components of the name field it does not itself directly access the

Christian Name and Surname Directories in the database: rather, it invokes module 'DW' to do this on its behalf. To explore the strategy further let us consider the handling of the surname 'WILSON'. When module 'N3' wishes to have the Surname Directory searched it achieves this by executing the following paragraph:

```
SEARCH-DIRECTORY.

    MOVE 'SE' TO DB-FUNCTION.
    MOVE DIRECTORY-IDENT TO DSC-DIRECTORY-IDENT.
    MOVE SB-STRING TO DSC-NAME-STRING.
    CALL 'DWDBSEARCH'  USING  DB-T-BLOCK.
```

The parameter 'DB-T-BLOCK' in the 'CALL' statement is used for communicating the various items of search information between the caller, in this case 'N3', and 'DW': it is also used by 'DW' to pass back to the caller the results of the search, such as the code values located in the appropriate directory. The three 'MOVE' statements preceding the 'CALL' statement are used to set up fields in 'DB-T-BLOCK' as follows:

1. DB-FUNCTION. This variable indicates to 'DW' which operation is required, the value 'SE' indicating that a search is required. Other permissible values would be 'OP' and 'CL', used respectively at the beginning and end of a source translation run to have the database opened and closed (18).

2. DSC-DIRECTORY-IDENT. This variable specifies for a search operation which directory or directories are to be searched. In this example it would be set to the value 'SN', indicating

that the Surname Directory is to be searched.

3. DSC-NAME-STRING. This variable is used to pass to 'DW' the name which is to be searched for: in the present example it would contain the string 'WILSON'.

Module 'R2STPROCESS' makes a similar call on 'DW', in this case to search for a relationship, e.g. 'HEAD', in the appropriate directory. The code for the 'SEARCH-DIRECTORY' paragraph in this module is identical to that contained in 'N3', illustrated above. However, in this case the variable 'DSC-DIRECTORY-IDENT' is set to the value 'RN', indicating that the Relationship Name Directory is to be searched.

It should be observed that in the calling interface to 'DW' there are no inbuilt assumptions about how the directories are actually organised in the database. Therefore, if a radical re-organisation were required this could be achieved merely by changing the single module 'DW'.

5.3.4 The Organisation of a Module's Interface to IDMS

In this section I shall examine the way in which IDMS is invoked by the use of Data Manipulation Language (DML) statements from within a COBOL module. As in previous sections I shall be concerned primarily with establishing methods which are consistent with a sound,

overall design strategy.

Even within a module which has a direct interface to IDMS, such as 'DW', only a fraction of the code is actually involved in making calls to IDMS: the remainder is concerned with interpreting the caller's request, preparing to make the call, unpacking and setting up the results after the call is executed, dispatching error messages, and so on. In an attempt to narrow even further the interface to IDMS it was therefore decided to isolate all the calls to IDMS in a special section of the module, which was given the name 'DB-ROUTINES SECTION'. Such a strategy can, for example, serve to assist the re-organisation of the module if a change in the method of accessing the database is desired.

In order to observe how the interface to IDMS is actually managed I shall continue further with the example provided in the previous section, and consider how 'DW' locates the surname 'WILSON' in the Surname Directory. A detailed description of the design and operation of the name directories will be provided in Chapter 7. For the purposes of the present discussion, therefore, I shall limit myself only to those aspects which are essential to an understanding of how 'DW' initiates the search. The Surname Directory, as described in the Schema, consists of 4 record types and 3 set types. However, module 'DW' needs only a limited view of the directory, since it is responsible merely for retrieval functions. Within the Source Translation Subschema, in fact, only two of the record types and one of the set types are present. This limited view of the Surname

Directory is illustrated in Figure 5.1.

The record 'N-SN-STRING' is a CALC-type record, each occurrence of which will contain a particular surname string. Thus, if 'WILSON' is present in the Surname Directory then there will exist a corresponding 'N-SN-STRING' CALC record for it. 'N-SN-ENTRY-SET' is, in practice, a 1:1 set, connecting each 'N-SN-STRING' record to its associated 'N-SURNAME' record. The latter record holds various items of information about the surname, and, in particular, the code value which is to be used for all subsequent processing by the linkage system.

Consider the actions which 'DW' must take to access the Surname Directory. These are as follows:

1. It must present the surname string value to IDMS with a request that the corresponding 'N-SN-STRING' CALC record should be located.

2. If IDMS reports that this record is not present in the database then 'DW' should dispatch a warning message to the user. For example, this could be a message such as:

   'WILSON' IS NOT PRESENT IN THE SURNAME DIRECTORY

3. If IDMS successfully locates the record then 'DW' should issue a further request to obtain the corresponding 'N-SURNAME' record. When this is obtained 'DW' can then pass on the

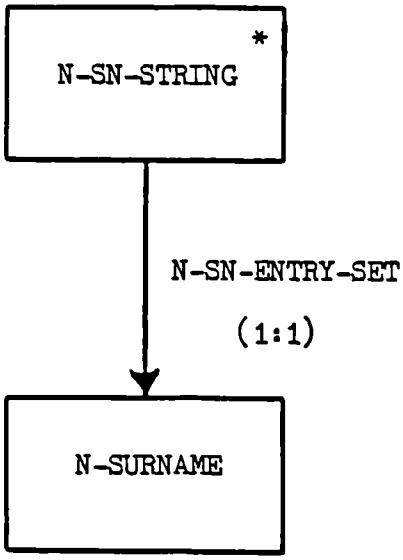


Figure 5.1 The 'view' of the Surname Directory as perceived by module DWDBSEARCH in the Source Translation Subsystem
(* = CALC record)

appropriate surname code value to the caller.

The method by which control is transferred to IDMS is precisely the same as the way that control is passed from one COBOL module to another, i.e. via a 'CALL' statement and parameters. Consider first the parameters. Within the WORKING-STORAGE SECTION of the DATA DIVISION of module 'DW' there is a special communication region, which is used to transmit control information between IDMS and the module. For example, a 4-byte field in this region, called 'ERROR-STATUS', is used by IDMS to indicate whether or not it has successfully carried out the operation which has been requested (19). Also in the WORKING-STORAGE SECTION are data declarations for the various database records which are to be accessed. For example, there are declarations for the records 'N-SURNAME' and 'N-SN-STRING', as follows:

```
01  SURNAME-RECORDS.
    02  N-SURNAME.
        03  N-S-SNAME-STRING.
            04  N-S-SN-CHARS      PIC X(20).
            04  N-S-SN-LGTH       PIC S9(2) COMP-3.
        03  N-S-UNIQUE-NUM.
            04  N-S-MAJOR-NUM.
                05  FILLER        PIC 9  VALUE 1.
                05  N-S-NUM       PIC 9(5).
            04  N-S-SYNONYM-IND PIC 99.
                88  MAJOR-SNAME VALUE 0.
        03  N-S-NORM-INIT         PIC X.
        03  FILLER                PIC X.
    02  N-SN-STRING.
        03  N-SNS-STRING          PIC X(20).
```

Such declarations establish buffer space in the module in which records being written to the database or being read from the database can temporarily reside.

I am now in a position to examine the code in 'DW' which actually causes IDMS to effect the retrieval of information from the Surname Directory. This code, which is to be found in the paragraph 'SEARCH-SN-DIRECTORY' within the DB-ROUTINES SECTION, is as follows:

```
       SEARCH-SN-DIRECTORY.

           MOVE DSC-NAME-STRING TO N-SNS-STRING.

       *   FIND CALC N-SN-STRING.
                       MOVE 0022 TO DML-SEQUENCE
                        CALL "ICL9IDMS" USING IDBMSCOM (32)
                                              SR163.

           IF  DB-REC-NOT-FOUND

               PERFORM SEND-NOT-FOUND-MESSAGE

           ELSE

               PERFORM IDMS-STATUS

       *       OBTAIN FIRST N-SURNAME WITHIN N-SN-ENTRY-SET
                       MOVE 0023 TO DML-SEQUENCE
                        CALL "ICL9IDMS" USING IDBMSCOM (18)
                                              SR161
                                              N-SN-ENTRY-SET
                                              IDBMSCOM (43);
               PERFORM IDMS-STATUS
               MOVE N-S-NORM-INIT TO DSC-NORM-INIT
               MOVE N-S-UNIQUE-NUM TO DSC-UNIQUE-CN-SN-NUM.
```

The first statement in this paragraph causes the surname string 'WILSON', as set up by module 'N3' in 'DSC-NAME-STRING', to be moved into the 'N-SN-STRING' record buffer. The next statement, 'FIND CALC N-SN-STRING', is a request to IDMS to search in the database for the record whose surname string value is currently in the 'N-SN-STRING' record buffer. This 'FIND' statement is not true COBOL, and were it to be presented to the COBOL compiler the compilation would fail. The statement is, in fact, a DML statement, and all such

statements must be translated into COBOL by a special IDMS preprocessing program before the module is submitted for compilation. The output from the preprocessing program is shown on the three lines following the 'FIND' statement: it will be observed that the single DML statement has been expanded into two COBOL statements, the second of which is a 'CALL' to invoke IDMS. In this 'CALL' there are two parameters, both of which identify fields in the special communication region. The first is used to specify the type of operation which is required, i.e. 'FIND CALC' in this case, while the second selects the particular record type which is to be used in the operation. Finally, it should be observed that the preprocessing program has converted the DML statement itself into a standard COBOL comment line by placing an asterisk in column position 7.

When IDMS returns control to 'DW' it will have set up 'ERROR-STATUS' to indicate the success, or otherwise, of the operation. Such an operation can have three possible outcomes, as follows:

1. The operation was successful, and the record requested has been found in the database.

2. The operation was unsuccessful, because the record could not be found.

3. The operation was unsuccessful, for some alternative and unpredictable reason (20).

The code following the call to IDMS must therefore be able to respond to each of these situations. The initial line, 'IF DB-REC-NOT-FOUND', deals with the second of the three alternative situations, and it enables an appropriate warning message to be dispatched to the user. If this condition has not occurred then the paragraph 'IDMS-STATUS' is invoked. This is a general-purpose error-trapping routine, which is invoked after all calls on IDMS, and after any expected error conditions (such as 'DB-REC-NOT-FOUND' in the present case) have been explicitly tested for. It interrogates the value of 'ERROR-STATUS', and if this indicates that there has been some unanticipated error it then proceeds to print out diagnostic information and abort the execution. If, however, it detects no error it allows the execution to continue.

Assuming that the 'N-SN-STRING' record for 'WILSON' has been successfully located the next step is to locate the corresponding 'N-SURNAME' record. This is achieved, as shown, by issuing the second DML statement, as follows:

```
OBTAIN FIRST N-SURNAME WITHIN N-SN-ENTRY-SET
```

As in the previous case it will be seen that the preprocessing program has converted the DML statement into a comment and replaced it by two COBOL statements, similar to the earlier ones. The design of the Surname Directory is such that for each 'N-SN-STRING' there must exist a corresponding 'N-SURNAME' record. Therefore, in the code which follows this second call to IDMS it is not necessary to include an

explicit check to confirm that the record has been found. As before, however, 'IDMS-STATUS' is invoked to trap any unanticipated errors. Assuming that the operation has been successfully completed IDMS will have retrieved the required 'N-SURNAME' record and placed it in the corresponding buffer space in the WORKING-STORAGE SECTION of 'DW'. All that now remains is for 'DW' to unpack the required information from this buffer (including the code value, stored in 'N-S-UNIQUE-NUM'), transfer it to corresponding fields in its 'DB-T-BLOCK', and return this and control back to 'N3STPROCESS'. The operations required to achieve access to the Surname Directory have been completed.

In this chapter I have examined the modular, 'top-down' design strategies which have been adopted in the development of the linkage system by considering the operation of a part of the Source Translation Subsystem. I have looked in some considerable detail at the way in which names in census household occupant records are handled, beginning with the modules which are responsible for scanning the source records, and moving progressively down to the module, 'DWDBSEARCH', which accesses the various name directories, via IDMS, in order to convert names into code values. The flow of control through these levels of the system is summarised in Figure 5.2. Once again, it should be emphasized that, as far as possible, each of the modules shown has been designed to correspond with some object (e.g. a household) or a function (e.g. a search of the name directories) which is meaningful in the 'real world'.

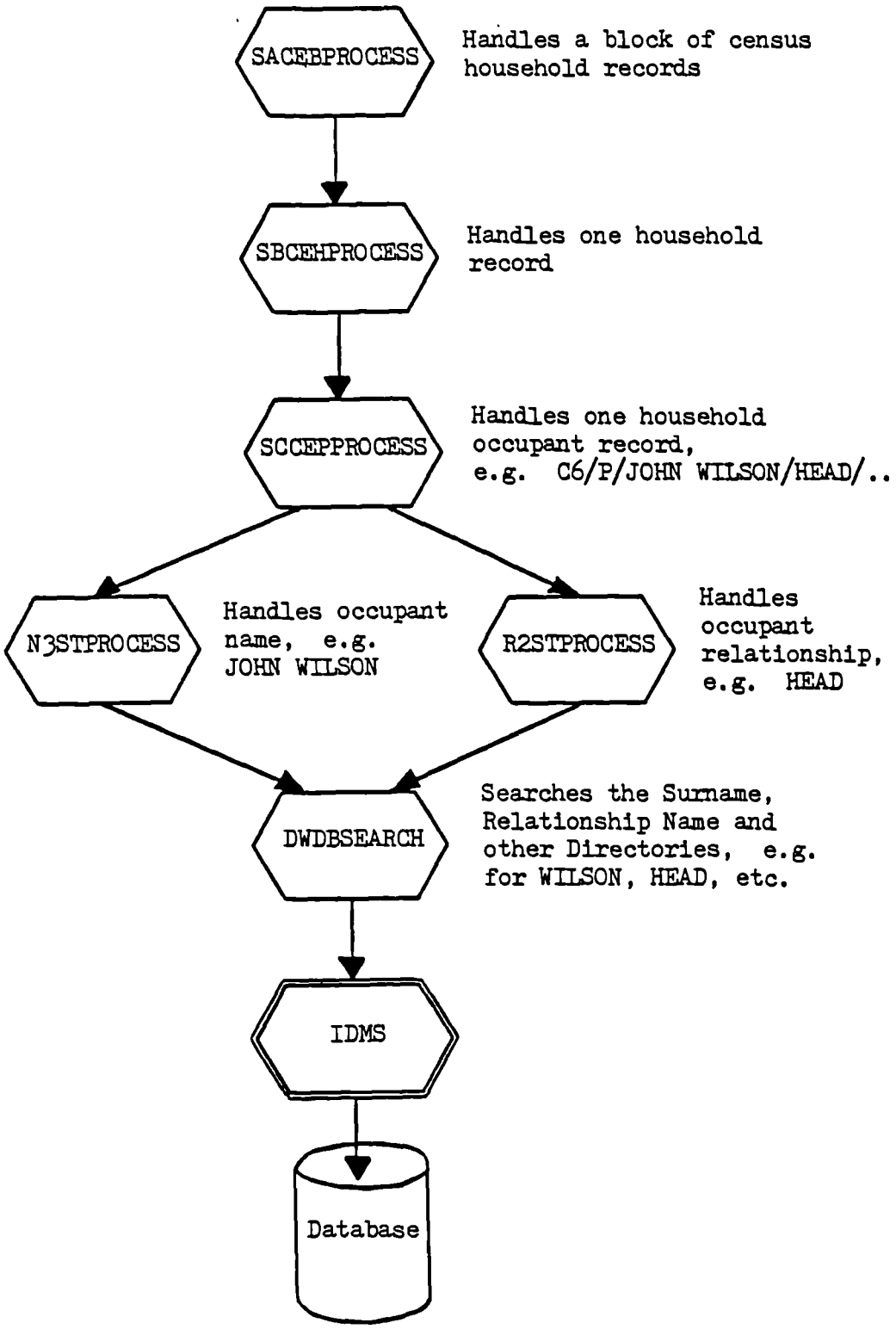


Figure 5.2 The structure and flow of control through a section of the Source Translation Subsystem

5.3.5 The Adoption of a Modular 'BIND' Philosophy

Earlier in this chapter, in Section 5.2.2, I drew attention to one of the penalties of the strategy of disaggregation, viz. the problem of ensuring that the separate parts of the system can ultimately be made to function as an integrated whole. An extension of this problem occurs when IDMS is used for complex applications, and it relates to the organisation of the record buffers which are used to provide communication between the program and the database. Since, in practice, this problem is capable of severely undermining the modular structure of the total system I feel that it is important for me to examine it in some detail and to present my preferred solution.

The subschema for the Source Translation Subsystem is a simple one, consisting of only 1 area, 10 record types and 5 set types. It was possible, therefore, to place all the code needed to interface with IDMS in the single module, 'DW'. By contrast, the subschema for the Record Linkage Subsystem has access to all 3 areas, 39 record types and 38 set types. Because the interface to IDMS is so much more complicated in this case it was necessary to disaggregate the code and set it up in as many as 27 individual modules. Of these modules, 3 handle the interface to the names and codes in the Directory Database, 15 the interface to the records in the Source Database, 7 the interface to the Population Database, and the remaining 2 modules have overall controlling functions. Given such a broad interface to IDMS, there could clearly be serious communication and control problems between the modules. To explore this difficulty further the

program-IDMS interface must be analysed in more detail.

After a database has been OPEN'd it is not possible for the program to access records in the database until it has carried out a series of 'BIND' operations. For example, for the record type 'N-SN-STRING', the appropriate DML instruction is:

```
BIND N-SN-STRING
```

The effect of executing this instruction is to make known to IDMS the location of the 'N-SN-STRING' record buffer in the WORKING-STORAGE SECTION. Subsequently, IDMS will know where it is to deposit an occurrence of this record coming from the database and where to collect an occurrence which is to be written to the database. Now where the interface to IDMS is distributed across several modules it is possible to adopt one of several alternative strategies for organising the BINDing of records.

At one extreme one can have a main control module in which reside the buffers for all the records in the subschema. The consequences of adopting such a policy are as follows:

1. When the database has been OPEN'd this module must arrange to execute a 'BIND' instruction for each of the record types.

2. Since other modules will need to access records and execute DML statements it will be necessary for the record buffers to

be passed to them via parameters in the 'CALL' statements. Where there is a large number of record types, for example there are 39 in the Record Linkage subschema, this can make inter-module communication very cumbersome.

3. With this type of organisation, where there is effectively a common pool of record buffers accessible to many modules, the task of redesigning a particular record can be troublesome. Since the record will be 'visible' in a large number of modules widespread recompilation will usually be necessary.

A more modular strategy, and the one that I have adopted within the present system, is to disaggregate the buffers along functional lines and to allocate them to several modules. The consequences of adopting this alternative policy are as follows:

1. When the database has been OPEN'd it is necessary to invoke several modules to execute 'BIND' instructions. In the Record Linkage Subsystem, for example, there are 11 such modules.

2. Since the record buffers can be declared within the functional areas of the system where they are to be used there is less need for extensive and cumbersome parameter passing between modules.

3. Where the redesign of a particular record is required this will affect only a limited area of the system, and very few

modules will need to be recompiled.

It should be evident that this more modular strategy towards the program-IDMS interface is consistent with the other programming and design strategies which have been described.

NOTES

1. This assumption may not be valid for computers at the smaller end of the range, and, in particular, for microcomputers. Here the 'universal' languages are more likely to be BASIC and PASCAL.

2. In the intervening period since the decision about database systems was taken there have, however, been significant advances in the technology which have served to increase the desirability of the relational option. In the first place, there is considerable standardisation, and the so-called Structured Query Language (SQL) is now an international standard for relational systems (ISO 1989). Secondly, the present draft version of SQL includes proposals for a 'recursive union' operation (ISO-ANSI 1989, 189-96). This facility, when implemented, will considerably simplify the kinds of inter-table manipulations which are required by record linkage systems. Finally, computer hardware is so much cheaper and faster than it was that the relative inefficiencies normally incurred in the relational approach are now of reduced significance.

3. Since August 1980, however, the VME/B operating system was replaced by EMAS, the Edinburgh Multi-Access System: this system could also support IDMS. For the reference manuals for the IDMS data description languages and data manipulation language see International Computers Limited 1977B and 1976A, respectively. For the 2980 VME/B reference manual see Edinburgh Regional Computing Centre 1977.

4. This version of COBOL is based closely on the ANSI 74 standard. (ANSI = American National Standards Institute) For a formal definition of the language see International Computers Limited 1977A and 1976B. For the programmers' guide see International Computers Limited 1978.

5. This version of FORTRAN is approximately equivalent to IBM FORTRAN IV (level G and H extended).

6. For example, it provides IF ... THEN ... ELSE ... and DO ... WHILE ... constructs for selections and iterations, respectively, and these constructs can be nested indefinitely. In addition, the PERFORM verb provides a convenient and simple means of executing a particular sequence of code, viz. a PARAGRAPH or SECTION: it is therefore essentially a mechanism for invoking a procedure for which there are no parameters.

7. There is one subsystem, the Input-Output Subsystem, which is an exception to this. It is invoked by each of the other subsystems to handle a number of common input/output, string-handling and error-reporting functions.

8. The policy adopted for naming modules is as follows:
   - the initial character is used to indicate the functional area with which the module is concerned. Thus, 'S' indicates source records, 'N' people's names, 'T' time (i.e. dates and ages), 'D' directories, and so on.
   - the second character is used to discriminate between all the modules within a particular functional area. The first two characters therefore provide a simple key which can be used to uniquely identify any module. (Such abbreviated forms are subsequently used for convenience in the thesis when making repeated reference to particular modules. For example, 'SC' is used as an abbreviated form of 'SCCEPPROCESS'.)
   - when the second character is a number or is a letter in the early part of the alphabet, e.g. as in 'SACEBPROCESS', this serves to indicate that the module is a 'pure COBOL' module, i.e. it does not interface with IDMS. Where the second character is a letter near the end of the alphabet, e.g. as in 'DWDBSEARCH', this indicates that the module does have an interface with IDMS. It is valuable to have the modules explicitly distinguished in this way since those which have an interface with IDMS cannot be submitted directly to the COBOL Compiler, but must be preprocessed by a special IDMS utility program. This operation is described more fully in Section 5.3.4.
   - the remaining characters of the name serve as a brief, mnemonic description of the module's main function.

9. All the data which is to be accessed by a COBOL module must be explicitly declared in a part of the module called the 'DATA DIVISION'. Within this division there are two major sub-areas: the 'WORKING-STORAGE SECTION' and the 'LINKAGE SECTION'. Where space is to be reserved in the module for constants and variables, appropriate declarations must be placed in the WORKING-STORAGE SECTION. Where the module needs to have access to parameters which have been passed from a calling module, then corresponding declarations for the parameters must be placed in the LINKAGE SECTION.

10. The word 'PIC' is short for 'PICTURE', and 'X' means any character. The term 'PIC 9(2)' describes a more restricted type of variable: again, it can hold two characters, but in this case each must lie in the range 0-9, e.g. '63'.

11. See Note 10.

12. The COBOL designation 'PIC S9(2) COMP-3' indicates that the variable is to hold signed integer values in the range -99 to +99.

13. The earlier versions of BASIC and FORTRAN, for example, were not well-suited to structured programming, but the more recent language specifications (e.g. FORTRAN 77) do incorporate improved facilities. The language which appears to have been most orientated towards structured programming is ALGOL 68.

14. The 'ELSE/NEXT SENTENCE' construct used in this conditional is effectively a means by which the end of the nested conditional can be explicitly indicated, in a situation where no action is required on the 'ELSE' selection. If the construct were omitted then the compiler would mistakenly assume that the following 'ELSE IF' and 'ELSE' selections were to be treated as part of the nested conditional, and the logic of the paragraph would be broken. This potential difficulty is graphically referred to as 'the dangling ELSE' problem.

15. Such a facility is often referred to as a 'macro' facility.

16. Where one wishes to set up a number (and possibly a large number) of files with similar characteristics then it can be both convenient and efficient to have them set up together as a single 'partitioned' file. As such, each member can be individually accessed when required. And at the same time, should one also wish to operate on the whole file, for example, to request that it should be archived to a tape, then such an operation is also permitted.

17. The Input-Output Subsystem operates only in association with the other subsystems, and it therefore does not require its own subschema.

18. The operations of 'opening' and 'closing' a database are very similar to those of opening and closing files in a traditional filing system. Thus, when the database is being opened it is necessary to indicate which database areas will be needed, and what types of access (e.g. read, write, etc.) to each area are required.

19. If the operation has been successful then the value '0000' is returned. If the operation has been unsuccessful then the value in 'ERROR-STATUS' can indicate why it was unsuccessful. For example, where there has been a request for a CALC record which cannot be located the value '0326' is returned.

20. This third category normally covers programming errors, such as where IDMS has been requested to carry out an operation which is not meaningful in the present context. A request to IDMS to find, say, the first member record in a set in a situation where no particular set has been previously identified would be an example of such an error.

CHAPTER 6 THE OVERALL STRUCTURE OF THE RECORD LINKAGE SYSTEM

Having devoted the last chapter to an examination of programming and database implementation issues I shall now turn my attention to the design features of the record linkage system which has been developed. It is intended that the approach which is adopted in this and the following five chapters will serve two distinct purposes. In the first place the method of presentation should provide a comprehensive analysis of the system, in terms of its overall structure and philosophy, and also of the way in which its individual parts are made to function. But, at the same time, it is intended that this system analysis will provide a convenient framework within which I can explore the fundamental methodological and design issues which are central to the development of record linkage strategies.

In the present chapter I shall be primarily concerned with the overall functional requirements of the record linkage system, and my emphasis will be on establishing what the system is required to do, rather than on how it is to accomplish it. In the following five chapters I shall then proceed to examine in detail the internal design of the system.

I shall begin by summarising the main functional requirements of a record linkage system. In the first place it is required to accept and process a variety of different nominal record types. Secondly, it

must be able to link these together in an appropriate fashion and on the basis of the name and other information contained in each record. Finally, it must provide suitable access to the linked data, in order that the results of the record linkage operations can be made available for analysis. As far as possible the user's interface to the system should be convenient and flexible, and the operational performance of the system, while not crucial, should be acceptable.

In earlier chapters, and, in particular, in Chapters 3 and 4, I have explored the nature of the problems of nominal record linkage and have sought to identify the kinds of solutions which might be appropriate. It is now necessary to consider the design of a complete system which can embody these 'solutions' and which is also able to satisfy the functional requirements which have just been summarised.

Figure 6.1 illustrates the overall scheme which has been devised. The diagram essentially portrays the results of the first stage in the 'top-down' design of the system, and it shows how the total process of record linkage can be disaggregated into a number of discrete steps. The most significant items in the diagram are the 'subsystems', represented by hexagonal boxes. Each of these subsystems is a major, autonomous functional component of the system, having its own set of responsibilities and operating in relative isolation from the others.

In addition to identifying the main functional components of the total system, Figure 6.1 is also intended to show how they 'fit together', i.e. how they interface with each other, the user and the

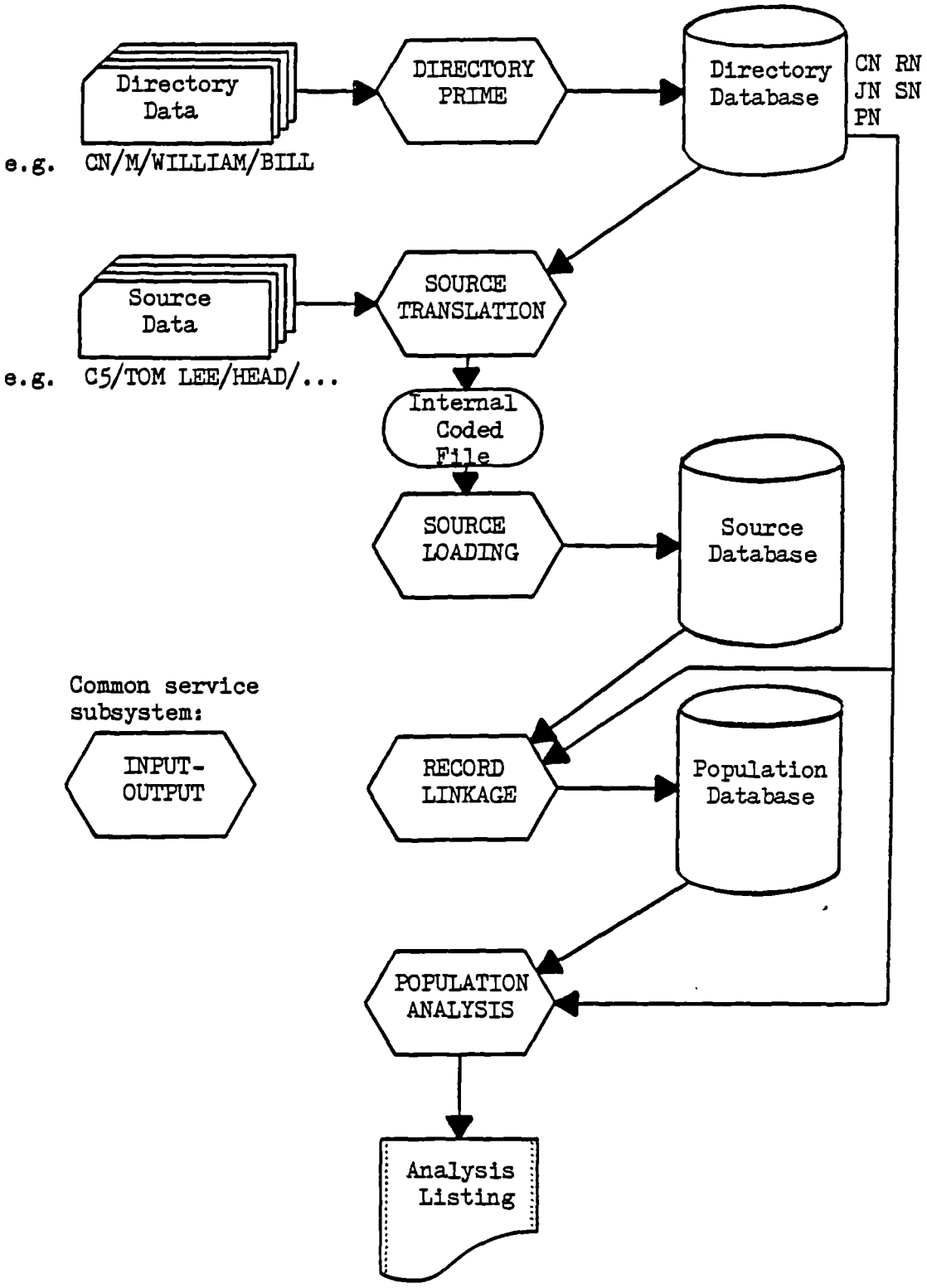


Figure 6.1 The overall organisation of the system and database

database. The connecting lines in the diagram represent the main information flows in the system, with the direction of each arrow indicating the direction of flow. If we consider the system as a whole, therefore, we can see that it has two major information inputs and one major output. The inputs, shown at the top left-hand side, consist of Directory Data (i.e. information about names) and Source Data; the output, shown at the foot of the diagram, consists of the Analysis Listing (1).

The remaining information flows in the diagram represent the 'internal' movement of information within the system, and the means by which the individual subsystems are able to 'co-operate' with each other and contribute to the total record linkage process. An important design feature in this respect is the way in which the subsystems use the database as a means of communicating with each other. Let us examine this feature more closely. The three major subdivisions of the database, represented by the cylinders on the right of the diagram, are the Directory Database, the Source Database and the Population Database. It will be observed that the subsystems do not pass information directly between themselves, but, rather, use each of the databases as a type of 'communication device' (2). Thus, typically, a subsystem will carry out its operations and then deposit the 'results' in one of the databases. The subsystem which is responsible for the next sequence of operations will then retrieve the 'results' from the database, and so the process continues. An example of this feature of the design is the role played by the Source Database in providing communication between the source handling

subsystems, shown in the upper part of the diagram, and the Record Linkage Subsystem. From this perspective it can thus be seen that a database is able to provide, not only the capacity for modelling a 'real world' view, but also, more practically, a valuable tool for integrating the separate functional components of a complex system.

In my analysis of the system thus far I have tended to place the 'processes', i.e. the subsystems, at the centre and to relegate the database to a peripheral and more subservient position. Indeed, this arrangement of priorities does, to an extent, reflect the dominant view which has been adopted during the system implementation, and which, as a result, has influenced the form in which I have presented the issues in this thesis. The origin of this view lies partly in the emphasis which has historically been placed on record linkage 'algorithms', but also in the considerable complexity which is involved in the implementation of the individual 'processes' themselves.

However, it is important to recognise that for applications which employ a database it is often more appropriate to position the database at the centre and to relegate the 'processes' to a more subservient role. Indeed, the transition towards a more database-centred view of record linkage has, in fact, taken place in the process of conducting the present research, and the consequences of adopting such a shift in emphasis will become apparent in the kinds of proposals for future work which are identified in later chapters (3).

In the remainder of this chapter I shall examine the functional role of each of the subsystems shown in Figure 6.1. My particular objective will be to establish the contribution which each subsystem is required to make to the overall process of record linkage and the nature of the interface which it presents to the user.

6.1 THE DIRECTORY PRIME SUBSYSTEM

In Sections 3.1 and 3.2 I examined the problems of nominal, spatial and occupational ambiguity, and established the need for providing name directory facilities. Essentially the requirement is that the system should provide the user with the appropriate means for submitting information about names, and it must subsequently be able to make use of this information during linkage operations. Typically, for example, the user may wish to inform the system that 'WILLIAM' is a male Christian name, and that 'BILL' is to be regarded as an acceptable synonym for it. Or, again, the user may wish to provide the information that 'DAUGHTER' is a female, kin relationship, and that 'DAUR' is to be regarded as an acceptable synonym for it.

In the present system design such information is held in the Directory Database (see Figure 6.1), and the Directory Prime Subsystem is the program which has responsibility for accepting the name information from the user and for organising it in the Directory Database. In this section I shall examine the characteristics of the

user interface and the way in which the name information is organised.

The Directory Database needs to contain five independent directories, holding information about the following types of names: Christian names, job-names (i.e. occupations), placenames, relationship names and surnames. The Directory Prime Subsystem is designed to provide the user with a range of facilities enabling him to organise information in the directories in a simple and convenient manner. Typically a user will wish to insert a batch of entries into a directory, but he may also subsequently wish to modify entries or even resubmit a complete directory. In addition, he may also wish to obtain an alphabetic listing of one or more directories. Facilities to assist the user with these operations are provided. They are as follows:

1. Database initialisation. When the Directory Database is created it must first have a number of initialising operations carried out on it. The user is able to initiate these operations simply by submitting the following command to the subsystem:

   DD/DBINIT

   The initial tag, 'DD', identifies this as a Directory Database command: tags 'SS' and 'PP' are similarly used to identify Source and Population Database commands, respectively. The string 'DBINIT' specifies the operation required (4).

2. Add entries. The 'ADD' command is used to request the subsystem to insert names into a directory. For example, if the user wishes to have a particular group of names inserted into the Christian Name Directory he requests this as follows:

```
DD/CN/ADD
CN/F/ADA/ADELAIDE
CN//ADAM
CN//ALBION
CN//ALEXANDER/ALEX'R
CN//ALFRED
CN/F/ALICE
CN/F/ANN/HANNAH/NANCY/ANNE
CN//ANTHONY
CN/F/AVERAL/AVERAIL
CN/F/BARBARA/BARBARY
CN/F/CAROLINE
CN/F/CATHERINE/KATHERINE/CATHARINE/KATHARINE
```

The 'DD' statement once again specifies the operation that is required, the code 'CN' indicating that additions are to be made to the Christian Name Directory (5). Each of the following lines uses the same code 'CN' as an identifying tag: this is included merely for checking purposes. Following the tag there is a field which indicates the gender to be associated with the Christian names which follow. This can contain the character 'M' (male), 'F' (female) or 'N' (null). Alternatively, the field can be left empty, in which case male will be assumed. The string or strings which follow, each separated by the character '/', constitute a synonym group of Christian names which is to be inserted in the directory and given appropriate code values. The first name has special significance, and is regarded, in a sense, as being the

'leader' of the synonym group: as such, it is called the 'major name'. If it is required to add further names to a group it is necessary merely to include these in a statement along with the major name. For example, to add 'ANNA' and 'ANNIE' to the 'ANN' synonym group it would be necessary merely to include the statement:

```
DD/CN/F/ANN/ANNA/ANNIE
```

3. Remove entries. Where limited reorganisation of a directory is required, such as the merging of two synonym groups, it is desirable to be able to remove entries first. This can be achieved by means of the 'REMOVE' command. For example, if the user wishes to remove the names 'HANNAH' and 'NANCY' from the Christian Name Directory he requests this as follows:

```
DD/CN/REMOVE
CN/HANNAH
CN/NANCY
```

Where the name nominated for removal is a major name the subsystem will treat this as a request for the removal of the whole synonym group associated with it.

4. Empty directory. Where radical reorganisation of a directory is required it is desirable to be able to empty the directory completely. This can be achieved by means of a command of the form:

```
DD/CN/EMPTY
```

5. Print directory. Where a user wishes to insert a new name into a directory he must decide whether to incorporate it as a new major name or else as a member of a synonym group which is already present. To assist the user with such an operation the subsystem can provide him with a complete alphabetic listing of any directory by means of the 'PRINT' command. For example, to obtain a listing of the Christian Name Directory the user issues his request as follows:

```
DD/CN/PRINT
```

The following is the initial section of the listing of such a directory:

```
          CN DIRECTORY
          ************

A
  ADA                              000001.00  F  A
      ADELAIDE                            01
  ADAM                             000002.00  M  A
  ADELAIDE                         ->
                   ADA
  ALBION                           000003.00  M  A
  ALEX'R                           ->
                   ALEXANDER
  ALEXANDER                        000004.00  M  A
      ALEX'R                              01
  ALFRED                           000005.00  M  A
  ALICE                            000006.00  F  A
  ANN                              000007.00  F  A
      HANNAH                              01
      NANCY                               02
      ANNE                                03
  ANNE                             ->
                   ANN
  ANTHONY                          000008.00  M  A
  AVERAIL                          ->
                   AVERAL
  AVERAL                           000009.00  F  A
      AVERAIL                             01
```

```
B
  BARBARA                        000010.00  F  B
      BARBARY                           01
  BARBARY                        ->
                    BARBARA

C
  CAROLINE                       000011.00  F  C
  CATHARINE                      ->
                    CATHERINE
  CATHERINE                      000012.00  F  C
      KATHERINE                         01
      CATHARINE                         02
      KATHARINE                         03
```

It will be observed that for each major name, e.g. 'ANN', there is an entry containing an 8-digit code value: this is the unique value which will be used in place of the name in all subsequent record linkage processing (6). There are also two alphabetic codes, the first being the gender indicator, already discussed, and the second the 'initial' character value which will be associated with each member of the synonym group. Thus, for example, if the system were presented with the two sets of Christian names 'MARY ANNE' and 'MARY HANNAH' it would register the initials as 'M A' in both cases. This can obviously be of value should the linkage system wish to compare initials. On the lines which follow a major name entry are details of the associated members of the synonym group. Alongside each member is a 2-digit code: this when added to the major name gives the code value for the member. Thus, for example, the code value for 'ANNE' is '000007.03'. Each such subsidiary member of a synonym group also appears for convenience in the main alphabetic list, together with a

cross-reference to the major name. Given a new name to be entered, therefore, a user can easily scan the directory to locate any potential synonyms which may already be present.

All five directories operate in approximately the same way, the only minor differences between them concerning the name attribute information, such as gender, which is submitted when entries are ADDed to the directories. The following are examples, one for each directory, of valid name entry statements:

```
SN/SIMPSON/SIMSON
CN/M/THOMAS/THOS/THO'S
PN/N/BUCKINGHAMSHIRE/BUCKS
JN/N/SERVANT/SERV/GENERAL SERV
RN/M/K/FATHER
```

Entries for the Surname ('SN') Directory are the simplest, in that they require no attribute information. Entries for the Job-name ('JN') Directory, like those for the Christian Name Directory, need to have associated gender information. In the example shown the gender is specified as 'N' (i.e. null), since a servant can be male or female. The attribute associated with placenames is used to indicate whether or not the place is within the base zone: this zoning technique was discussed fully in Section 3.2.2. In the example shown the attribute value 'N' indicates that Buckinghamshire is not in the base zone. Finally, entries for the Relationship Name ('RN') Directory must be supplied with two attribute values. The first

specifies gender in the usual way, while the second indicates whether the relationship is a kin relationship or not. In the example shown 'Father' is presented as a male, kin relationship.

There is one significant difference between the Relationship Name Directory and the others. In the case of all the other directories the code values can be allocated to the names in a quite arbitrary way, since the system does not need to attach any particular significance to the names themselves. However, for relationships the names do have significance, and when record linkage is subsequently being carried out the processing will need to be quite different where a relationship is specified as 'son', rather than 'brother', say. In consequence, therefore, the code values must be allocated in such a way that the system can correctly associate them with the corresponding relationships. This is achieved by insisting that the user inserts the more common relationships into the directory in a special order immediately after the directory has been created. The required sequence begins as follows:

```
DD/RN/ADD
RN/N//HEAD
RN/F//WIFE
RN///SON
RN/F//DAUR
```

The adoption of this sequence will ensure that 'HEAD' is allocated the code value '000001.00', 'WIFE' the value '000002.00', and so on. When the linkage system subsequently comes to use the relationship codes it can therefore test explicitly for these values (7).

6.2 THE INPUT-OUTPUT SUBSYSTEM

The system needs to make extensive use of input and output operations. All the input and output traffic between the programs and the database is, of course, handled by the IDMS system itself. But for all the other input and output traffic, i.e. that between the programs and the user, a specially implemented subsystem is used, viz. the Input-Output Subsystem. This localising of the responsibility for input and output operations in a single subsystem is a sound, modular design feature, one of the benefits of which is the simplification of the logic within the other subsystems.

The Input-Output Subsystem supervises all aspects of the processes concerned with the information transfer between the user and the system. Its main responsibilities are as follows:

1. Command and data input. All input which is submitted by the user is handled by the subsystem. Thus, it deals with each subsystem command, such as 'DD/CN/ADD', directory data input and primary source input. It arranges to open and close the input device at the start and end of a run, respectively, and to fetch each record. At a more detailed level it also arranges to control the scanning of each record and to extract the individual strings. Where a particular record extends over several lines it handles the detection of the continuation codes and the repositioning of the record scanning pointer.

2. Output listing. All information which is output to the user is channelled through this subsystem. In a typical Directory Prime run, for example, it arranges to open and close the line printer output device and to control the output of the following information:
   - 'banner headlines' at the start and end of the run, indicating, for example, 'START OF DIRECTORY PRIME' and 'END OF DIRECTORY PRIME'.
   - the user's command and data input.
   - any information explicitly requested by the user, e.g. the listing of a name directory.
   - warning or error messages, together, where appropriate, with an indication of where in the user's command or data statement the fault was detected.
   - at the end of the listing, either a 'SUCCESSFUL RUN' message or details of the number of warnings and/or errors which have been reported.

6.3 THE SOURCE TRANSLATION SUBSYSTEM

In Section 3.3 I examined the problems of 'data capture', i.e. the task of transferring information from a variety of source documents and setting it up in the computer in a form which will facilitate its subsequent processing. The first part of the problem concerns the form in which the user presents each record for input,

and in Section 3.3.2 I established the characteristics of some suitable record input formats, i.e. formats which permit the data to be input with minimum effort and maximum accuracy. Unfortunately, formats which are designed to be especially convenient for the user will not generally be the most suitable for subsequent processing and record linkage (8). There is therefore a requirement to carry out a number of transformations on the input data in order to prepare it for subsequent manipulation, and it is the function of the Source Translation Subsystem to carry out these transformations. In addition, the subsystem is also required to subject the input data to a number of consistency checks, in order to establish whether it has been entered correctly.

The position of the Source Translation Subsystem in relation to the other components of the system is shown in Figure 6.1. It will be observed that the subsystem has two types of input, viz. the actual Source Data, as presented by the user, and also appropriate name information and codes from the Directory Database. Its output consists of the 'transformed' data, in the form of an Internal Coded File: this file will subsequently be input and loaded into the database by the Source Loading Subsystem.

The most important actions of the Source Translation Subsystem are as follows:

1. It takes as its main input a sequential file containing source records and produces as output a sequential file containing a

recoded version of the source. Data items in the input file (e.g. names and ages) can have a variable length and can appear in a variety of forms: data items in the output file have a fixed length and form.

2. It accesses the Directory Database to convert all Christian names, job-names, placenames, relationship names and surnames into their corresponding code values.

3. It replaces the marital status field by a single character code, with values as follows:

   U - unmarried

   M - married

   W - widow

   R - widower

4. It converts all dates into a special 'computed date' form. This is obtained for a particular date by calculating the number of days between the date and a reference date, viz. 31 December 1649. (9) For example, the population household census in 1851 was held on 30 March: the computed date value corresponding to this is 73502. (10)

5. It similarly converts all ages into a computed date of birth form. For example, where someone in the 1851 census was reported as having an age of two years, this would mean that his actual age was greater than or equal to two years and less

than three years. One would therefore be able to conclude that he was born at some time during the period 31 March 1848 and 30 March 1849. His date of birth could therefore be stated as 28 September 1848 +- 183 days, and the computed date of birth form for this would be 72589 +- 183.

6. For baptism records which do not contain a date of birth or age it estimates the date of birth from the date of baptism and the probable number of days which would have elapsed between the birth and the baptism. This interval can be set for a given batch of baptisms by the user (11).

7. Where there are ditto signs in a record it arranges, where possible, to make use of values from a previous record.

8. When scanning the details pertaining to a particular person it carries out consistency checks on the associated gender information. If, for example, it found a wife called William, or a widow who was a schoolmaster, it would issue a warning. Such warnings would prompt the user to check that the original source record had been read and entered correctly.

9. When scanning the details of the people in a household it carries out consistency checks on the information provided. For example, it checks that the age differences between the generations are sensible, and also that parents and children (excepting married daughters) have the same surname. Where

there is an inconsistency a warning message will be issued. Again, such warnings can prompt the user to check that the original source record has been read and entered correctly.

Let us now examine the form in which the user is required to present source records to the Source Translation Subsystem. The following example illustrates how a batch of baptism records from the Elwick Hall parish register (DCRO 1978 Vol 2, 22-3) would be entered:

```
SS/PR/BAP/B1/ELWICK HALL PARISH/ANG/30/EXTRACT

B1/14 MAR 1813/HENRY/S/RALPH/ISABELLA/ALCOCK/POPLAR ROW/FARMER
B1/13 APR/ELIZABETH/D/THOMAS/JANE/KAY/PLAINTREE HILL/FARMER
B1/17 JULY/CARTER/S/WILLIAM/MARY/CROWE/NEWTON HANSARD/FARMER
B1/5 DEC/THOMAS/S/ROBERT/ELIZABETH/SWALWELL/AMMERSTON HILL/FARMER

B1/8 MAR 1814/ANN/D/THOMAS/ANN/WILSON/MIDDLE STOTFOLD/FARMER
B1/8 APR/WILLIAM/S/JOHN/MARGARET/DOBING/HOLE HOUSE/BLACKSMITH
B1/5 JUNE/WILLIAM/S/GEORGE/SUSANNA/ROBINSON/AMMERSTON HALL/FARMER
B1/12/ANN/D/ROBERT/JANE/LANCHESTER/POPLAR ROW/FARMER
B1/14 JULY/ANN/ID//ISABELLA/DOBSON/HIGH STOTFOLD/SINGLE WOMAN
B1/14 AUG/JOHN/S/JOHN/MARY/BROWN/BURNTOFT/FARMER
B1/23 DEC/ANN ELIZABETH/D/RALPH/ISABELLA/ALCOCK/POPLAR ROW/FARMER
```

The initial 'SS' statement provides information about the source records which are to follow. Thus 'PR' specifies parish register records; the code 'CE' would, alternatively, specify census records. 'BAP' indicates that the records relate to baptisms, while 'B1' defines the format in which the records are presented, as was discussed earlier in Section 3.3.2. The next field contains the name of the parish, and 'ANG' identifies the denomination as Anglican. The number '30' indicates the birth-baptism interval in days which is to be applied when birthdates are being calculated. Finally, 'EXTRACT'

is essentially just a documentary comment, indicating that only a section of the register is included.

Each of the following lines contains information about one baptism, the initial format identification code 'B1' being included merely for checking purposes. The details of the layout of these records was discussed fully in Section 3.3.2. The only minor, additional point to be made here is that blank lines may be inserted anywhere among the source lines, for example, at the end of each year, to improve the presentation. Appendix B contains additional examples of input code formats.

There are several other design features of the Source Translation Subsystem which are intended to assist the user in his operations. For example, the file which is presented for translation is not restricted to holding just one type of source, e.g. 'B1' baptisms. Rather, it can contain many types: baptisms, marriages, census, etc. This flexibility enables the user to maintain his input source files in the way which is most convenient for him.

In addition, the subsystem can be made to operate in a variety of modes. At the simplest level it can be used merely to check a block of data, without the production of any code. This is particularly useful where a new primary source is being introduced, and it is necessary to find out what additional names will need to be incorporated in the name directories. Once the directories have been suitably updated then a full code production run can be carried out.

6.4 THE SOURCE LOADING SUBSYSTEM

The output from the Source Translation Subsystem is a sequential file containing nominal records in an internal coded form. As such, it will be considerably simpler for the system to handle than the original source records as input. However, were it to be organised in precisely this form for record linkage then there could still be significant problems involved in using it. In order to carry out a particular record linkage operation, for example, it might be necessary to select a small number of records from such a sequential file and link them with selected records from other such files. With files that are organised in a purely sequential fashion this record selection can be accomplished only by reading through all the records in each file, and this can clearly be a time-consuming operation.

In order to avoid such operational inefficiency it is necessary to transfer the records to a random-access storage device and to provide appropriate name indexing facilities. Alternatively, given the use of a database system, one can achieve the same goal by storing the records in a database. Within the present system design, therefore, I have included a suitably structured 'Source Database' into which I arrange to load the internal coded nominal records. And to control the associated loading operations I have provided a corresponding subsystem, viz. the Source Loading Subsystem. The organisation of these components is shown in Figure 6.1.

In comparison with source translation the process of loading source requires little interaction with the user. Only minimal facilities are therefore needed. These are as follows:

1. Database initialisation. Before any code is loaded into the Source Database it must have a number of initialising operations carried out on it. The user is able to initiate these operations simply by submitting the following command to the subsystem:

   SS/DBINIT

   This corresponds with the command 'DD/DBINIT', which is used to initialise the Directory Database.

2. Database load. In order to cause a file of code to be loaded into the database it is necessary only for the user to issue the command:

   SS/DBLOAD

   The file can contain the whole of the code which is to be inserted in the Source Database; alternatively, the user can maintain the code in several files (e.g. baptisms in one, marriages in another, etc.) and arrange to load these individually.

6.5 THE RECORD LINKAGE SUBSYSTEM

The actual process of linking nominal records is accomplished in the present system by transferring data from the Source Database and restructuring it afresh in the Population Database. A preliminary discussion of the fundamental structural properties of a population database was provided in Section 4.3.3.

In view of the present, exploratory nature of family-based record linkage work it is regarded as essential that the user of a record linkage system should be given considerable control over the linkage operation. Were a system, for example, to provide no control and to permit only the linkage of the entire contents of the Source Database then it would allow little scope for studying the processes of record linkage and for exploring alternative linkage strategies. If, instead, the user is able to specify, for example, that only records of a particular type are to be linked, say just parish register records, then this can enable him to carry out a comparative evaluation of the results of using alternative source types. What, for example, would be the characteristics of a population database created only from parish register records, and how would this compare with a population database created only from census records?

In view of the possibility that the linkage strategies may cause erroneous connections to be made it is also regarded as essential that the user should be provided with suitable facilities for monitoring and evaluating the reliability of the linkage process. He should, for

example, be able to initiate a limited linkage run, for which he would select both the records to be linked and also the order in which the linkages are to be made. In addition, to enable him to carry out his own manual linkage as a check against the system he should also be able to have displayed the particular records which participate in any linkage operation.

The subsystem which has responsibility for providing the facilities described above is the Record Linkage Subsystem. As shown in Figure 6.1 it is required to take information directly from the Source Database and transfer it to the Population Database. It also retrieves information from the Directory Database, which it uses for record display purposes. Since the records which it handles are in internal coded form, each code value needs to be translated back into its corresponding, original character string form prior to display. To accomplish this the subsystem makes appropriate accesses to the name directories in the Directory Database.

The main facilities provided by the Record Linkage Subsystem are as follows:

1. Database initialisation. As for the Directory and Source Databases it is necessary for the Population Database to be initialised. This must be done before any linkage operations are carried out. The appropriate user command is:

   PP/DBINIT

2. Source type selection. The user is able to specify that only certain types of the records in the Source Database are to be linked. For example, the command:

PP/SOURCES/C5/B1

would permit only 1851 census records and 'B1' format baptism records to be used in any subsequent linkage operations. Other permissible variants of this command are as follows:

PP/SOURCES/PR

PP/SOURCES/CE

PP/SOURCES/ALL

The first command would allow only parish register records to be linked, and the second only census records. The final command, which is the default, allows all records to be used.

3. Link and print data. The user is able to request that records only for a particular surname are to be linked, and he can additionally request that a listing of them is also to be produced. For example, to have the records for 'SMITH' printed and linked he would issue a command of the form:

PP/PRINTSOURCES,LINK/SMITH

Note that this command would operate only on the types of records specified by the previous 'PP/SOURCES' command, and that it would locate records for 'SMITH' and for any synonyms of 'SMITH'. An example of the use of such a command is illustrated in Figure 6.2. In this case five records have been located for the surname 'BONE'. In the display listing it will be observed that an asterisk is used to highlight each

```
:PP/PRINTSOURCES,LINK/BONE

BONE
****

1861 CENSUS HOUSEHOLD      C00051      18  ELWICK HALL PARISH          (B)     SUNDERLAND COTTAGE
**********************
  * WILLIAM BONE              M  HEAD          MAR  FORESTER              6 OCT 1804 +- 183D  WOLVISTON        (P)
  * JANE BONE                F  WIFE          MAR  WIFE                  7 OCT 1809 +- 183D  SEDGEFIELD       (N)
  * MARGARET BONE            F  DAUR          UNM  SCHOLAR               7 OCT 1850 +- 183D  WOLVISTON        (P)

1871 CENSUS HOUSEHOLD      C00076       6  ELWICK HALL PARISH          (B)     SUNDERLAND COTTAGE
**********************
  * WILLIAM BONE              M  HEAD          MAR  WOODMAN               1 OCT 1803 +- 183D  WOLVISTON        (P)
  * JANE BONE                F  WIFE          MAR                        2 OCT 1809 +- 183D  SEDGEFIELD       (N)
  * SARAH J BONE             F  DAUR          UNM                       30 SEP 1844 +- 183D  ELWICK HALL      (B)
  * WILLIAM BONE             M  GRAND SON                                1 OCT 1868 +- 183D  ELWICK HALL      (B)

1861 CENSUS HOUSEHOLD      C00040       7  ELWICK HALL PARISH          (B)     MIDDLE STOTFOLD
**********************
    ROBERT PATTISON          M  HEAD          MAR  FARMER                7 OCT 1819 +- 183D  ELWICK           (B)
    FRANCES PATTISON         F  WIFE          MAR  WIFE                  7 OCT 1822 +- 183D  STOTFOLD         (B)
    ANN PATTISON             F  DAUR          UNM  SCHOLAR               7 OCT 1850 +- 183D  ELWICK           (B)
    JOHN PATTISON            M  SON           UNM  SCHOLAR               6 OCT 1852 +- 183D  STOTFOLD         (B)
    FRANCES PATTISON         F  DAUR          UNM  SCHOLAR               7 OCT 1853 +- 183D  STOTFOLD         (B)
    ROBERT PATTISON          M  SON           UNM                        7 OCT 1855 +- 183D  STOTFOLD         (B)
    EDWARD PATTISON          M  SON           UNM                        7 OCT 1857 +- 183D  STOTFOLD         (B)
    MARY E PATTISON          F  DAUR          UNM                        7 OCT 1859 +- 183D  STOTFOLD         (B)
    GEORGE ATKINSON          M  SERV          UNM  FARM SERV             7 OCT 1831 +- 183D  OSMOTHERLY       (N)
    THOMAS DUCK              M  SERV          UNM  FARM SERV             6 OCT 1820 +- 183D  ELLERBY          (N)
    THOMAS DAWSON            M  SERV          UNM  FARM SERV             6 OCT 1844 +- 183D  TRAPSTON         (N)
    MARTHA BARTON            F  SERV          UNM  FARM SERV             7 OCT 1841 +- 183D  WOLVISTON        (P)
  * MARY A BONE              F  SERV          UNM  NURSE                 7 OCT 1846 +- 183D  WOLVISTON        (P)
```

```
1870 MARRIAGE M2 ANG        M00073          ELWICK HALL PARISH          (B)          2 JUL 1870    BNS
**************
    RICHARD REED                    LONG NEWTON                (N)   BACH   SIG    BORN  1 JAN 1850  +- 183D
                                                                                  HUSBANDMAN
                                    FATHER:    WILLIAM REED                       CORDWAINER
  * MARGARET BONE                   ELWICK HALL PARISH         (B)   SPIN   SIG    BORN  1 JAN 1851  +- 183D
  *                                 FATHER:    WILLIAM BONE                       WOODMAN

1868 BIRTH    B1 ANG        B00402          ELWICK HALL PARISH          (B)      SUNDERLAND LODGE          (B)
**********
  * WILLIAM BONE                    M    BAP  24 OCT 1868    BORN  24 SEP 1868  +- 30D      *ILLEGIT*
                                    FATHER:    -
  *                                 MOTHER:    SARAH BONE                         @@@@@@@@
```

Figure 6.2 The linking and printing of records for a particular surname by the Record Linkage Subsystem

occurrence of the nominated surname and that computed dates of birth are printed, rather than ages. In this form the display can therefore considerably assist the user, should he wish to link the records manually. Mention should also be made of the record identifiers which serve to identify each record uniquely: thus, for example, the first census record listed has been given the identifier 'C00051'. As will be demonstrated later such identifiers can be used to cross-reference the records when inspecting Population Analysis listings.

The user is provided with additional record selection options. For example, if he wishes to link records exclusively for the name 'SMITH', and not include its synonyms, then he is able to request this as follows:

```
PP/PRINTSOURCES,LINK/SMITH/EXCLUSIVE
```

or, more tersely, in the form:

```
PP/PRI,LIN/SMITH/EXC
```

Other permissible examples of this command are as follows:

```
PP/PRI,LIN/A
PP/PRI/ALL
PP/LIN/ALL
```

The first command would cause the records for all surnames beginning with the letter 'A' to be printed and linked (12). The second command would print the records for all

surnames: these would be printed in alphabetical order by surname. Finally, the third command would link, but not print, the records for all surnames.

6.6 THE POPULATION ANALYSIS SUBSYSTEM

Having linked a number of nominal records and stored them in a Population Database it is then necessary to provide some means for accessing the linked data and retrieving for the user the information which he requires. Ideally, a user would wish to be able to obtain a suitable response to any ad hoc query that he chooses to make, e.g. 'Obtain frequency distributions of age-at-death for farmers and for agricultural labourers'. Additionally, the user would probably find it valuable if the system could arrange to set up selected subsets of the data from the database in files capable of being processed in a sophisticated way by statistical analysis packages, such as SPSS (Statistical Package for the Social Sciences). (Nie et al 1975)

Within the constraints of the present research and of the type of database management system which has been used it has not been practicable to provide the kinds of facilities just described. It was decided, therefore, that a more limited, but still valuable, objective would be the provision of analysis facilities geared towards the monitoring of the linkage operation. It was considered that a more user-oriented interface could be incorporated, as necessary, at a

later stage.

The subsystem responsible for providing analysis facilities is the Population Analysis Subsystem, and its position in relation to the other components of the system is shown in Figure 6.1. Like the Record Linkage Subsystem it is required to translate internal coded values back into their corresponding, original character string forms prior to display. It therefore needs to make similar accesses to the name directories in the Directory Database.

The main function of the Population Analysis Subsystem is to display details about selected groups of people and families in the database. The specification of the precise information to be displayed is under the direct control of the user, and there are two types of selection available to him. Firstly, he is able to select the amount of information which he requires about each 'case', i.e. a person or a family, in which he is interested. And secondly, he is able to specify the characteristic(s) of the cases to be selected. The facilities which are provided are as follows:

1. Detail level selection. Within the database there can be a considerable amount of information about any one individual or family. The user must therefore be provided with some means for selecting how much information is to be displayed, and the corresponding mechanism for controlling this is the 'DETAILLEVEL' command. If, for example, the user submits the command:

PP/DETAILLEVEL/PERSONS=MINIMUM/FAMILIES=MINIMUM

then an absolute minimum of information will be produced. Thus, for each person only the name, gender, date of birth, birthplace and the person's unique identification key, as generated by the system, will be produced. Likewise for each family only the names of the marriage partners, the marriage details and the family's unique identification key will be produced. If, alternatively, the user submits the following command:

PP/DETAILLEVEL/PERSONS=MAXIMUM/FAMILIES=MAXIMUM

then considerably more information will be produced. For each person there will additionally be details of the parents, marriage partners, number of children and each record in which the person was mentioned. While for each family there will additionally be comprehensive details of the parents and each child. Between these two extremes it is possible to make a number of intermediate selections. For example, the command:

PP/DET/PER=PARENTS/FAM=CHILDREN

would for each person provide additionally only the details about parents, while for each family it would provide additionally only the details about children.

2. Person and family subset selection. From all the persons and families present in the database the user should be able to select the particular ones in which he is interested. There are a number of ways in which he can do this. For example, by issuing the command:

```
PP/SUBSET=PERSONS/COHORT=1831-1840/B/M
```

the user is able to identify the group of males born in the base zone in the period 1831-40. Other permissible variants of this command are as follows:

```
PP/SUB=PER/COH=ALL/N/F
PP/SUB=PER/SURNAME=SMITH
PP/SUB=PER/NAME=JOHN SMITH
PP/SUB=PER/JOBNAME=SHOEMAKER
PP/SUB=PER/JOBNAME=SHOEMAKER/C5
```

The first command identifies all females not born in the base zone. The second command identifies all people having the surname 'SMITH', or a synonym of it, while the third identifies all people having the name 'JOHN SMITH', or a synonym. The fourth command identifies all people who at some point had the occupation 'SHOEMAKER', or a synonym, and the final one all people who in the 1851 census had the occupation 'SHOEMAKER', or a synonym.

Subsets of families can be specified in a similar fashion. For example, the command:

```
PP/SUBSET=FAMILIES/COH=1830/ALL
```

identifies those families for which the marriage was celebrated in 1830. Other permissible variants are as follows:

```
PP/SUB=FAM/SURNAME=SMITH
PP/SUB=FAM/COUPLE=JOHN,ANN SMITH
```

The first of these commands identifies all the 'SMITH'

families, while the second identifies the subset of the 'SMITH' families for which the marriage partners are called 'JOHN' and 'ANN'.

3. Print person and family subsets. Having identified a particular subset of persons or families in the database the user would then normally wish to have some operation carried out on each member of the subset: this operation would typically involve analysis or presentation of the data. The subsystem, as currently implemented, provides only the presentation facility, and this can be activated by issuing the following command as the first command in the complete sequence:

```
PP/PRINT=ON
```

The effect of issuing this command is that each subset of persons or families which is subsequently identified by means of a 'SUBSET' command will be automatically printed (13). If the facility were to be generalised to allow additional operations to be carried out on each selected subset then other, similar commands could be provided.

Examples of the printing of person and family subsets are illustrated in Figures 6.3 and 6.4 respectively. In the first example the requirement is to obtain comprehensive information about all the people in the database with the name 'WILLIAM BONE'; while in the second the requirement is to obtain comprehensive information about each 'BONE' family. It should be noted that these examples correspond with the record

```
:PP/DETAILLEVEL/PERSONS=MAXIMUM/FAMILIES=MAXIMUM
:PP/SUBSET=PERSONS/NAME=WILLIAM BONE

WILLIAM BONE
************

   P00031    WILLIAM BONE                    M     6 OCT 1804  +- 183D  WOLVISTON          (P)
   ******

      FATHER -              NOT KNOWN
      MOTHER -              NOT KNOWN

      MARRIAGES -
         FAMILY F00007:     WIFE    = P00032    JANE @@@@@@@@                  7 OCT 1809  +- 183D  SEDGEFIELD          (N)
                               SONS = 0    DAUGHTERS = 2

      EVENTS -
         1861 CENSUS:        7 APR 1861  HEAD          MAR  FORESTER            6 OCT 1804  +- 183D  WOLVISTON          (P)
                             C00051       18  ELWICK HALL PARISH        (B)      SUNDERLAND COTTAGE
         1871 CENSUS:        2 APR 1871  HEAD          MAR  WOODMAN             1 OCT 1803  +- 183D  WOLVISTON          (P)
                             C00076        6  ELWICK HALL PARISH        (B)      SUNDERLAND COTTAGE

   P00035    WILLIAM BONE                  I M     1 OCT 1868  +- 183D  ELWICK HALL          (B)
   ******

      FATHER -              NOT KNOWN
      MOTHER -              P00034    SARAH J BONE                30 SEP 1844  +- 183D  ELWICK HALL          (B)

      MARRIAGES -
         NONE

      EVENTS -
         1868 BAPTISM:       24 OCT 1868                                      24 SEP 1868  +-  30D
                             B00402       ANG  ELWICK HALL PARISH        (B)      SUNDERLAND LODGE          (B)
         1871 CENSUS:        2 APR 1871  GRAND SON                             1 OCT 1868  +- 183D  ELWICK HALL          (B)
                             C00076        6  ELWICK HALL PARISH        (B)      SUNDERLAND COTTAGE
```

Figure 6.3 The printing of a person subset by the Population Analysis Subsystem

```
:PP/SUBSET=FAMILIES/SURNAME=BONE


BONE
****
****


WILLIAM & JANE BONE
*******************

    F00007    WILLIAM BONE                  P    MARRIAGE:  DETAILS UNKNOWN
    ******    JANE @@@@@@@@                 N

        FATHER -              P00031     6 OCT 1804  +- 183D  WOLVISTON          (P)
        MOTHER -              P00032     7 OCT 1809  +- 183D  SEDGEFIELD         (N)

        CHILDREN -
            P00034:           SARAH J BONE                F   30 SEP 1844  +- 183D  ELWICK HALL      (B)
            P00033:           MARGARET BONE               F    7 OCT 1850  +- 183D  WOLVISTON       (P)
```

Figure 6.4 The printing of a family subset by the Population Analysis Subsystem

linkage operation illustrated in Figure 6.2, and it is to be recommended therefore that a detailed, comparative examination of the two displays should be carried out. During such an examination it will be clearly observed how the inclusion of record identifiers, e.g. 'C00051' and 'C00076', in the 'EVENTS' sections of Figure 6.3 serves to cross-reference the person details to the original nominal records presented in Figure 6.2. The way in which the individual records have been linked can therefore be readily monitored. In a similar fashion the use of unique person and family identifiers in Figures 6.3 and 6.4 serve to distinguish and cross-reference the people and families described. For example, the two people called 'WILLIAM BONE' are clearly distinguished by their associated unique identifiers, 'P00031' and 'P00035'.

Finally, at a more detailed level, it will be observed that there is a problem when displaying a woman's name, in that one can choose to use either her maiden or a married name. In the present system this is resolved by always listing the woman's maiden surname: where the maiden surname is not known it is replaced by the string '@@@@@@@@'. (14) The use of this string is illustrated in Figures 6.3 and 6.4, in the printing of the name of the wife of 'WILLIAM BONE' (P00031) as 'JANE @@@@@@@@'.

4. Print statistics. As the user builds up structures in the Population Database he may wish to obtain some brief statistical information about the contents of the database,

such as how many people and families are present and how many names are represented. He can obtain this by issuing the command:

PP/STATISTICS

Typical output is as follows:

```
POPULATION DATABASE STATISTICS
******************************

PERSONS        =  1563
FAMILIES       =   322
SURNAMES       =   310
PERSON NAMES   =  1213
FAMILY NAMES   =   283
```

I have now completed my analysis of the overall design of the system and of the functional role of each of its subsystems. In the next chapter I shall begin my examination of the more detailed, internal design of the system and of the strategies which are employed.

NOTES

1. There are a number of other types of input and output information, which, for simplicity, have been omitted. Thus, for example, most of the subsystems operate under the control of commands which are submitted by the user. In addition, there are facilities which enable the contents of the Directory and Source Databases to be displayed. Further details about these additional types of information are provided later in the chapter.

2. The only exception to this is the method of transferring internal coded source data between the Source Translation and Source Loading Subsystems. Since such data is used only in a strictly sequential way there is no advantage to be gained from storing it in the database. In this case the appropriate 'communication device' is an ordinary sequential file.

3. The proposal in Section 10.3.2 to use person and family occurrence templates reflects this shift in emphasis, as does also the proposal in Section 11.4 to employ a dual arrangement of IDMS and a relational database system. Concluding remarks on this change in the overall design perspective are provided in Section 13.1.2.

4. In all subsystem commands keywords, such as 'DBINIT', can be abbreviated, for convenience, to the first three characters. An alternative form for the command for having the Directory Database initialised would therefore be:

   DD/DBI

5. In the example shown the group of names to be added is submitted in alphabetical order. This is, however, not a requirement, and the names can be added in any convenient order.

6. The method used for allocating the code values was described in Section 3.1.2. In general, the user does not need to be aware of the existence of such values: they are created and manipulated internally by the system. However, they can be of interest to a user who wishes to monitor the internal workings of the system. It should be added that the decimal point displayed in each code value does not have any mathematical significance, but is used only to distinguish that section of the code, viz. the final two digits, which discriminates the individual members of a synonym group.

7. A significant advantage of having the linkage system test code values rather than the actual names is that it facilitates the use of the linkage system to handle records written in some foreign language. For example, to handle French census records one would merely need to present the appropriate relationship names to the directory as follows:

```
DD/RN/ADD
RN/N//CHEF
RN/F//FEMME
RN///FILS
RN/F//FILLE
  etc.
```

Assuming only that the names were presented in the appropriate sequence the linkage system would be able to operate correctly, irrespective of the original source language.

8. For example, in Section 2.2.2, I drew attention to the difficulties of handling date information. While a person entering data may find it natural and convenient to present dates in their normal calendar form, e.g. '1 JULY 1989', the computer system would find it simpler to handle a serial day number, e.g. '32689'.

9. The criterion for the selection of a reference date is that it should be sufficiently early to pre-date any date which might at some time need to be handled by the system.

10. Although such computed date calculations use 31 December 1649 as the reference date it should be noted that they do not take into account the switch to the Gregorian Calendar, which took place in 1751-2. It would only be necessary, however, to make an appropriate correction for this if records from around this period, or earlier, were being used.

11. From an observation of the occasional entries where a date of birth was specified the user may be able to provide a reasonable estimate of the birth-baptism interval which was customary during the associated period.

12. This variant of the command would, for example, permit the user to create a 'mini' population database, to be used, perhaps, for preliminary, experimental purposes. By making an appropriate selection of initial letters he would be able to create a population database whose size matched his analytical requirements.

13. For convenience in the present implementation a default setting of 'PRINT=ON' is assumed, and so the inclusion of each 'SUBSET' command will automatically result in a corresponding display of information.

14. The '@@@@@@@@' string is used in other situations to indicate missing information. For example, in the listing of the birth record in Figure 6.2 it serves to indicate that the parent's occupation was not specified.

CHAPTER 7 IMPLEMENTATION OF THE DIRECTORY PRIME SUBSYSTEM

In the last chapter I examined the overall design of the record linkage system, directing particular attention towards its functional requirements and user interface. I also set out to establish the contribution which each of the major functional components of the system, i.e. the 'subsystems', makes to the overall process of record linkage. This analysis has essentially provided us with a description of the first stage in the 'top-down' design of the system. In this and the following four chapters I move on to explore the next stage in the design process, i.e. a consideration of the problems associated with the design of each subsystem. In these chapters I shall be concerned with the nature of the functions which the subsystems are required to carry out, and the way in which such functions can be progressively disaggregated and ultimately implemented as a number of discrete, COBOL program 'modules'. Because these chapters are, of necessity, concerned with detailed design and implementation matters it is suggested that the reader should consider omitting them, or else skimming over them lightly, on an initial reading.

My method of analysing the system at the more detailed subsystem and module level will aim to follow two lines of investigation in parallel. On the one hand I shall be considering the structural design of the database; and on the other I shall be considering the functional requirements and design of the programs which are to access

the database (1). As stated in the last chapter, however, my constant aim will be to use this analysis as a framework in which to explore the fundamental methodological and design issues involved in the development of family-based record linkage strategies.

The bulk of the present chapter will be concerned with the provision of facilities for handling name information. I shall initially consider the design of the name directories and the way in which they may be held in the Directory Database. I shall then examine the nature of the processes which are involved in creating a directory, transferring user-supplied name information to it, and arranging for it to be displayed. I shall also examine the design of the program which has responsibility for executing these processes, viz. the Directory Prime Subsystem. Finally, at the end of the chapter I shall look briefly at the way in which the system is provided with a range of input and output facilities by the Input-Output Subsystem.

7.1 DESIGNING A DIRECTORY

Let us initially consider the fundamental properties of a directory. A directory is a repository of information, which is organised in such a way that any required item of information can be accessed readily. When designing a directory, whether of the computer or manual variety, one must therefore analyse its functional

requirements so that the information can be organised so as to satisfy these requirements.

Let us consider, as a simple example, the way in which information is organised in a telephone directory. Here the requirement is that a user, supplied with a person's name and address, should be able to obtain their telephone number. The directory is therefore structured so as to satisfy this requirement, and this is done by placing all entries in alphabetical order by surname. For all the people with the same surname, entries are ordered alphabetically by their initials, and for all those with identical surname and initials, entries are ordered alphabetically by street name. The access to a particular person's telephone number can therefore be rapid. However, if alternatively one wished to use a telephone directory to find the name and address of the person with a particular number then one would need to restructure the directory so that all items were placed in order by telephone number. For this kind of access an alphabetic ordering would be totally inappropriate, and many hundreds of pages would usually need to be scanned to locate a particular number.

By a similar token, when designing the name directories for a record linkage system one must consider carefully how they will be used and what the functional requirements are. In order to explore these matters in detail I shall, for the present, restrict my attention to the handling of Christian name information and to the design of the associated Christian Name Directory. The handling of

other types of information (e.g. surname) is very similar, and so will require little additional comment.

In Section 6.1 I examined the role of the Directory Prime Subsystem and looked in detail at the user's interface to the Christian Name Directory. What I shall now analyse are the precise access requirements which the Christian Name Directory must satisfy: once I have identified these I shall then be in a position to propose a suitable directory structure.

My starting point in this analysis is the fundamental requirement that for each Christian name in the directory there is to be a unique numerical code value, together with gender and 'normalised' initial character attributes, as described in Section 6.1. Further, the numerical codes are to contain a 'major code' prefix, which will permit the names to be organised into synonym groups. If it is assumed that by some means such information can be installed in the directory, the next step is to identify the ways in which the information will subsequently need to be retrieved. The following are considered to be the major requirements:

1. Given a Christian name obtain its code value and other attributes. For example, given the Christian name 'CATHARINE' the directory should be able to furnish the code value '000012.02', together with the gender attribute 'F' and the normalised initial character 'C' (2). This type of access will be required when a source record is being translated into

its internal coded form by the Source Translation Subsystem. It will also be used by the Population Analysis Subsystem when it is required to convert into internal coded form Christian names which appear in user commands, for example, as in:

PP/SUBSET=PERSONS/NAME=CATHARINE BELL

2. Given a code value obtain the corresponding Christian name. This is the converse of the first requirement. Thus, for example, given the code value '000012.02' the directory should be able to furnish the Christian name 'CATHARINE'. This type of access will be required whenever information from the Source or Population Databases is to be displayed for the user. It will be employed, for example, by the Record Linkage Subsystem when displaying source records in response to a 'PP/PRINTSOURCES' command.

3. Provide a complete traversal of the Christian names in the directory in alphabetical order. This type of access will be needed when the user requests the Directory Prime Subsystem to display the complete directory.

4. Given a non-major Christian name obtain the major Christian name in its associated synonym group. For example, given the Christian name 'CATHARINE' the directory should be able to furnish the corresponding major name 'CATHERINE'. This type of access will be needed by the Directory Prime Subsystem in order to create cross-references between non-major and major

names when displaying the complete directory (3).

5. Given a major Christian name provide a traversal of the Christian names in its synonym group. This type of access will be needed when displaying the complete directory, and it will also be needed when the user has furnished a particular major Christian name and requested that all members of its synonym group are to be removed from the directory.

In summary, it should be clear that the access requirements which are to be satisfied by the Christian Name Directory are considerably more elaborate than those associated with, say, a telephone directory. Not only must it be able to traverse all names in alphabetical order and provide efficient random access via any name, it must also provide efficient random access via any code value and arrange for the placing of names into synonym groups.

7.2 THE DIRECTORY DATABASE STRUCTURE

The natural starting point in the design of the Christian Name Directory is to propose that for each unique Christian name there should exist a corresponding Christian name record: this would hold all the information about the name, such as its internal code value and gender. If it were necessary merely to gain access to this record given the Christian name then this could be done by nominating the

record as a CALC-type record, and the design of the directory would be complete. However, since there are other access requirements to be satisfied it therefore becomes necessary to consider the introduction of additional, appropriate record and set types.

Essentially each Christian name record must exist simultaneously in two separate groupings. In the first, the alphabetic grouping, it must exist at its appropriate position when all the records are placed in alphabetic order by Christian name. In the second, the synonym grouping, it must exist within the cluster of records which form the synonym group to which it has been allocated. These two requirements would suggest that the Christian name record should participate as a member record in two distinct set types. The final access requirement, viz. that it should be possible to gain access to any Christian name record when given its internal code value, would suggest the use of another CALC-type record.

The complete record and set structure which has been developed to provide the required Christian Name Directory facilities is illustrated in Figure 7.1. As will be observed there are five different kinds of records and three different kinds of sets involved. In order to explore the design I shall consider the characteristics of each of the record types in turn:

1. N-CHRISTIAN-NAME. This is the most significant record type in the directory, and it holds all the information which is known about a particular Christian name. Thus, for example, for the

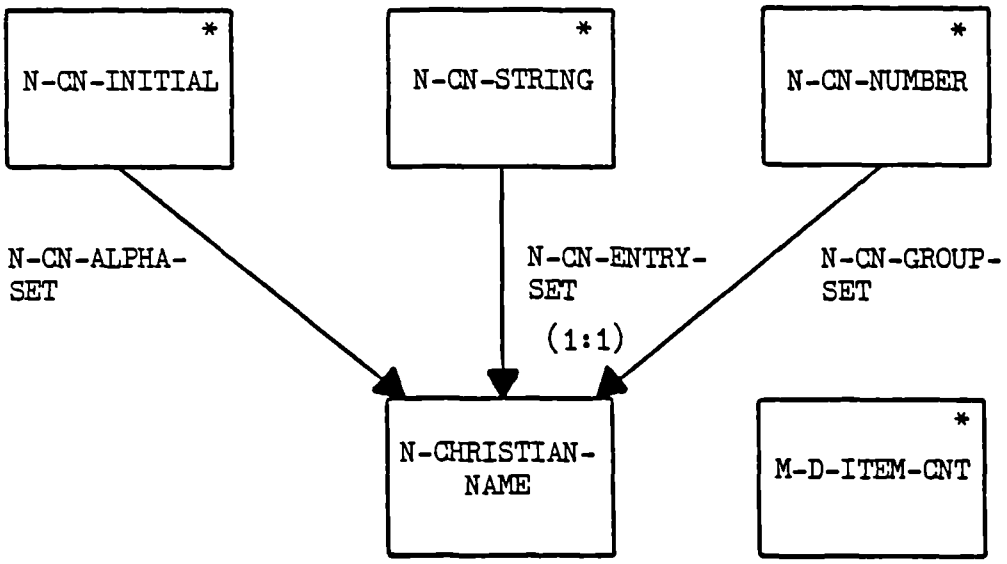


Figure 7.1 A complete set diagram of the Christian Name Directory
(* = CALC record)

name 'CATHARINE' it would hold the following four data items:

the name itself, 'CATHARINE'

the corresponding internal code value, '000012.02'

the appropriate gender indicator, 'F'

the normalised initial character value, 'C'.

2. N-CN-INITIAL. This record type provides for the alphabetic grouping of Christian names within the directory. When the Directory Database is initially created 27 occurrences of this record need to be set up, one for each letter of the alphabet, and the remaining one for all other characters. The function of this record is to provide access to all the Christian names beginning with a particular letter, and this is achieved by means of the associated 'N-CN-ALPHA-SET'. Each 'N-CHRISTIAN-NAME' record will be connected via the 'N-CN-ALPHA-SET' to the appropriate 'N-CN-INITIAL' record: for example, the 'CATHARINE' record will be connected to the 'C' record. A schema DDL option enables any set type to be defined as 'SORTED'. By using this option for the 'N-CN-ALPHA-SET' one can ensure that the 'N-CHRISTIAN-NAME' records will be maintained in strict alphabetical order within the set. Thus, for example, when the 'CATHARINE' record is presented for connection to the 'C' record IDMS will search along the chain of member records in the relevant set and find the appropriate insertion point. When the time comes to print the complete directory, therefore, initial presorting of the entries is unnecessary.

As indicated the 'N-CN-INITIAL' record is a CALC-type record, and it therefore provides an access route into the directory. In turn one can access the records for each of the letters of the alphabet and use the 'N-CN-ALPHA-SET' to locate all the names which begin with the corresponding letter. The 27th. occurrence of the 'N-CN-INITIAL' record is used to provide access to any name which does not begin with an alphabetic character (4).

An alternative way of organising this access route would be to have all the 'N-CHRISTIAN-NAME' records as members in a single set, with a single CALC-type record as owner. Unfortunately, this arrangement could become very unwieldy for IDMS to maintain, especially where the set was large. Each new member record would need to be inserted at its appropriate alphabetic position in the set, and this could involve a large number of disc transfers. The present arrangement, with the use of 'N-CN-INITIAL' records, keeps the sets reasonably small and the updating efficient.

3. N-CN-STRING. This record, also a CALC-type record, provides the second access route into the directory, in this case via the Christian name string. For example, if 'CATHARINE' is present in the directory then there will exist a corresponding 'N-CN-STRING' CALC record for it. 'N-CN-ENTRY-SET' is a 1:1 set, connecting the 'N-CN-STRING' record to its associated 'N-CHRISTIAN-NAME' record (5). Therefore, having entered the

directory via the name 'CATHARINE' one can obtain the corresponding code value and other information from the 'N-CHRISTIAN-NAME' record.

4. N-CN-NUMBER. This is also a CALC-type record, and it provides the third access route into the directory, in this case via an internal code value. At the same time it provides convenient access to the members of a synonym group. There will exist an 'N-CN-NUMBER' record for each major Christian name in the directory, and the function of the associated 'N-CN-GROUP-SET' is to connect to this record all the 'N-CHRISTIAN-NAME' records for the members of its synonym group. Consider, for example, how this access route may be used to obtain the name which corresponds with the code value '000012.02'. One can first use the 'CALC' facility to locate the 'N-CN-NUMBER' record which corresponds with the major number '000012.00' (i.e. for 'CATHERINE'). It is then a simple matter to traverse the 'N-CN-GROUP-SET' until the record for '000012.02' (i.e. for 'CATHARINE') is located.

Since it is possible to update the directory by adding further synonyms into any synonym group, each 'N-CN-NUMBER' record is designed to hold the two-digit code suffix which is to be used to construct the internal code value for the next synonym. For example, after adding 'CATHARINE', code number '000012.02', the two-digit code suffix in 'N-CN-NUMBER' would be set to '03'.

5. M-D-ITEM-CNT. The above four record types and their associated sets will satisfy all the accessing requirements of the directory. What is needed finally, in addition, is some means for providing a unique internal code value at the point where a new major name is being inserted in the directory. For example, when 'CATHERINE' was originally added the value '000012' was somehow provided. It is the function of the 'M-D-ITEM-CNT' record to furnish the next available code value to be used, and in the Directory Database there needs to be one occurrence of this record for each of the directories. When the Directory Database is set up the five records are created and are given initial values of '1'. After a unique internal code value has been allocated the appropriate 'M-D-ITEM-CNT' record is incremented by '1'.

Given this arrangement of different record types the process of entering a new Christian name into the directory can clearly be a somewhat complicated operation. If the Christian name is a new major name then it will be necessary to create three new records ('N-CHRISTIAN-NAME', 'N-CN-STRING' and 'N-CN-NUMBER') and also to access the appropriate 'M-D-ITEM-CNT' record to obtain a unique value for the internal code. Where a synonym is being added only two new records will be created ('N-CHRISTIAN-NAME' and 'N-CN-STRING'), but it will also be necessary to gain access to the 'N-CN-NUMBER' record for the appropriate major Christian name.

In addition to creating the various records it will also be necessary to connect them together within the sets shown. However, this will present no problem, since the connections are automatically carried out by IDMS. The descriptions of the three sets in the schema all specify that the membership of the 'N-CHRISTIAN-NAME' record is to be 'MANDATORY AUTOMATIC'. The term 'MANDATORY' indicates that no 'N-CHRISTIAN-NAME' record can exist in the database without being a member of the specified set; the term 'AUTOMATIC' indicates that when an 'N-CHRISTIAN-NAME' record is created IDMS will automatically arrange to connect it into the specified set (6).

In this study of the structure of the Directory Database I shall now consider the slight variations which need to be incorporated in the design of the five directories. In the last chapter we saw that the attribute information which needs to be associated with a particular type of name is dependent on the type of the directory. At the simplest level are surnames, for which there are no attribute values; while the most complex are relationship names, for which there are two attribute values, gender and the kin/non-kin indicator. Such variations, however, have little impact on the overall design of the directories: they merely constrain the way in which the central 'NAME' record must be structured to accommodate the particular attribute values.

Finally, it is necessary to consider a distinctive feature which I have incorporated into the design of the Surname Directory. The need arises because of the potential dual role of a surname in a

person's name, i.e. it can be present as a genuine surname, or else it can appear as a forename. For example, in the name 'HARRISON SMITH' the person's true surname is 'SMITH', and the surname 'HARRISON' acts as a pseudo-Christian name. This ambiguity in the use of surnames presents a difficulty for the design of the Christian Name and Surname Directories. The central question is: presented, for example, with the occurrence of a name such as 'HARRISON SMITH', should one arrange to place the surname 'HARRISON' in the Christian Name Directory? Although this would be a feasible solution it does seem to be rather pragmatic and inelegant, and it could lead to a significant increase in the size of the Christian Name Directory. A more appropriate strategy, and the one which has been adopted, is to employ the Christian Name Directory to hold only Christian names and the Surname Directory to hold only surnames (7). With this arrangement it then becomes necessary, when presented with a particular forename, to search for it first in the Christian Name Directory and then in the Surname Directory.

The adoption of this alternative strategy also demands that the permissible ranges of code values for Christian names and surnames must be arranged to be disjoint. If they were not, then when presented with a code value for a forename it would often be possible to translate it back into either a Christian name or a surname, and so there would be a consequent ambiguity. In the present scheme this problem is resolved by ensuring that all surname codes have the numeric digit '1' placed in the most significant position. Thus, for example, '100069.00' is the internal code for the surname 'HARRISON',

while '000069.00' is the code for the Christian name 'AGNES'.

7.3 THE OPERATION OF THE DIRECTORY PRIME SUBSYSTEM

So far in this chapter I have been concerned with the design of directories suitable for handling name information, and have examined a scheme for organising them in a Directory Database. I shall now address attention to the nature of the processes which are involved in manipulating the contents of the Directory Database.

The Directory Prime Subsystem is that part of the total system which is designed to load and maintain information in the five name directories. In order to do this it must supervise two quite separate functions, as follows:

1. the reading and checking of the user's commands and data, and the initiation of appropriate actions.

2. the actual carrying out of the required actions on the Directory Database.

I shall examine the nature of these functions in the following two sections.

7.3.1 Processing the User's Commands

Since each of the major subsystems is required to handle commands and/or data submitted by the user it is sensible to employ a common mechanism to supervise this. In the present system this common mechanism is implemented within the Input-Output Subsystem. It provides for the reading in of each record, the unpacking of individual fields from the record and the handling of fields which continue from one physical record to the next.

Let us consider briefly how the Directory Prime Subsystem makes use of these facilities. In order to obtain a record from the user it invokes the Input-Output Subsystem as follows:

```
MOVE 'FR' TO IB-FUNCTION.
CALL 'F2INPUT'  USING  INPUT-BLOCK.
```

The code 'FR' is set up in the field 'IB-FUNCTION', within 'INPUT-BLOCK', to indicate that a Fetch Record operation is required. When the operation is completed the Input-Output Subsystem will have obtained the record and stored it in a buffer in 'INPUT-BLOCK'. It will also have set a flag in 'INPUT-BLOCK' to show that the operation was completed successfully.

In order to obtain each string from the record the Directory Prime Subsystem makes successive calls on the Input-Output Subsystem as follows:

```
MOVE 'FS' TO IB-FUNCTION.
CALL 'F2INPUT' USING INPUT-BLOCK.
```

The code 'FS' in this case indicates that a Fetch String operation is required. When the operation is completed the Input-Output Subsystem will have obtained the next string from the record and placed it in a special string buffer within 'INPUT-BLOCK'. Where a string extends from one physical record to the next the Input-Output Subsystem arranges to read in the next record and continue the scanning. The subsystem also makes syntactic checks on the command and data records.

Given that the Input-Output Subsystem handles the inputting and 'unpacking' of the records and strings the Directory Prime Subsystem merely has to carry out a number of simple semantic checks on the input and then initiate the required actions on the Directory Database.

7.3.2 Accessing the Directory Database

In its response to the user's commands the subsystem must carry out appropriate transactions involving the Directory Database. It must cause new records to be created where necessary, and at other times it will need to cause records already present to be destroyed. It will also need to traverse various sets in the database, for example where the contents are to be displayed.

In order to examine the programming strategy employed within the Directory Prime Subsystem I shall consider three fundamental directory tasks: initialising the Directory Database, adding a Christian name and synonym, and listing the Christian Name Directory. A sequence of user commands which would initiate such tasks is as follows:

```
DD/DBINIT
DD/CN/ADD
CN/F/ANN/HANNAH
DD/CN/PRINT
```

The first operation, the initialisation of the Directory Database, involves the creation of the various records which are needed to support all the other directory operations. These include for each directory an 'M-D-ITEM-CNT' record and the set of 27 'initial' records. Since the user could inadvertently attempt to initialise the directory a second time some means of checking for and avoiding this should also be provided.

The precise actions which take place when the Directory Database is initialised are as follows:

- Firstly a check is made to ensure that the Directory Database is a new one. Part of the initialisation involves the creation of a special 'M-D-ITEM-CNT' record, which acts as a 'marker' record. Before any attempt is made to initialise a database, therefore, the existence of the marker record is checked: if it does exist then the initialisation is curtailed, and an error message is sent to the user. A

similar protection mechanism is used for the Source and Population Databases.

- The special 'M-D-ITEM-CNT' marker record is created.

- For each of the five directories an 'M-D-ITEM-CNT' record is created, containing the initial value '1'. These are the records which are subsequently used to furnish internal code values when new major names are added to the directories.

- For each of the five directories 27 'initial' records are created, one for each letter of the alphabet, together with the extra one to be used for names not beginning with an alphabetic character.

The actions which take place when a new major name, 'ANN', and a synonym, 'HANNAH', are added to the Christian Name Directory are rather more intricate. For the new major name it is necessary to check that there is a code value available, and for each name it is necessary to check that it is not already present. If it is decided that the addition is to proceed then corresponding 'N-CHRISTIAN-NAME' and 'N-CN-STRING' records must be created for each entry, and for the major name a new 'N-CN-NUMBER' record must also be created.

The detailed actions are as follows:

- A check is made to determine whether the code value in the 'M-D-ITEM-CNT' record for Christian names has reached its limiting, maximum value (8): if it has then an error message is sent to the user and the run is aborted.

- A check is made to determine whether the name 'ANN' is already present in the directory. This can be achieved by attempting to create an 'N-CN-STRING' record for 'ANN'. If the record already exists then IDMS reports this back by setting the special value '1205' in the 'ERROR-STATUS' field. In this case the Directory Prime Subsystem sends a warning to the user, and goes on to deal with the synonym 'HANNAH'.

- Assuming that the name 'ANN' was not present in the directory then the creation of the 'N-CN-STRING' record proceeds normally. A code value for 'ANN' is then obtained from the appropriate 'M-D-ITEM-CNT' record, and an 'N-CN-NUMBER' record for the new synonym group is created. Within this record the special two-digit code suffix, to be used when creating the first synonym code value, is initialised to '1'. The 'M-D-ITEM-CNT' record is incremented by '1'.

- The first character of the name 'ANN', i.e. 'A', is then used to locate the corresponding 'N-CN-INITIAL' record via the 'CALC' mechanism. Finally, an 'N-CHRISTIAN-NAME' record is

set up and created using the appropriate string, 'ANN', the newly allocated internal code value, the initial character and the gender code. Immediately IDMS has created this record in the database it will automatically connect it as a member in the three sets: 'N-CN-ALPHA-SET', 'N-CN-ENTRY-SET' and 'N-CN-GROUP-SET' (see Figure 7.1). The name 'ANN' is now fully installed in the Christian Name Directory.

- The first action when attempting to install the synonym 'HANNAH' is to check whether it is already present. This is carried out in precisely the same way as for 'ANN', i.e. by attempting to create the corresponding 'N-CN-STRING' record. If it is found to be present the Directory Prime Subsystem sends a warning to the user, and proceeds to deal with the next synonym, major name or command, as appropriate.

- Assuming that the name 'HANNAH' was not present in the directory then the creation of the 'N-CN-STRING' record proceeds normally. When inserting a synonym it is necessary to locate the 'N-CN-NUMBER' record for the corresponding major name. This can be achieved by first locating the 'N-CN-STRING' record for 'ANN': this is carried out using the normal 'CALC' mechanism. From this record it is possible to use the set mechanism within 'N-CN-ENTRY-SET' to locate the 'N-CHRISTIAN-NAME' record for 'ANN'. Finally, by extracting the major code value from this record and using the 'CALC' mechanism it is possible to obtain the 'N-CN-NUMBER' record

for 'ANN'. The special two-digit code suffix in this record enables an internal code value for 'HANNAH' to be constructed.

- The remaining directory operations are identical to those which were carried out for the major name. The first character of the name 'HANNAH', i.e. 'H', is used to locate the corresponding 'N-CN-INITIAL' record. And finally an 'N-CHRISTIAN-NAME' record for 'HANNAH' is set up and created. As before IDMS will automatically connect the newly created record into the three sets, and the updating operation is complete.

I shall finally consider the actions which need to take place in order to produce an alphabetic listing of the complete Christian Name Directory (9). In this case it is necessary to use the 27 'N-CN-INITIAL' records to provide access to every 'N-CHRISTIAN-NAME' record in the database. Synonym cross-referencing information is additionally obtained by making use of 'N-CN-NUMBER' records and the 'N-CN-GROUP-SET'.

The detailed actions are as follows:

- The letters of the alphabet are taken in turn, and for each letter the letter is printed, and then the following actions are carried out.

- The corresponding 'N-CN-INITIAL' record is obtained via the 'CALC' mechanism and the members of the 'N-CN-ALPHA-SET' are then accessed in turn: each of these 'N-CHRISTIAN-NAME' records represents a Christian name beginning with the corresponding initial.

- Where the 'N-CHRISTIAN-NAME' record is the record for a major name the system arranges to print out an initial entry for the name, including the appropriate internal code and other attribute values, followed by an entry for each of the synonyms. It does this by first using the internal code value stored in 'N-CHRISTIAN-NAME' to obtain the corresponding 'N-CN-NUMBER' record via the 'CALC' mechanism. It then traverses the 'N-CN-GROUP-SET to locate the 'N-CHRISTIAN-NAME' records for the synonyms.

- Where the 'N-CHRISTIAN-NAME' record is the record for a synonym the system arranges to print the name of the synonym, together with a cross-reference to the corresponding major name. It obtains the details of the major name by using the internal code value stored in 'N-CHRISTIAN-NAME' to obtain the corresponding 'N-CN-NUMBER' record via the 'CALC' mechanism. It then locates the first member of the 'N-CN-GROUP-SET', which, by definition, is the 'N-CHRISTIAN-NAME' record for the major name.

The tasks of removing entries from a directory and emptying a complete directory operate in a similar, though reverse, fashion. Thus, the removal of an entry is achieved by destroying the records which were originally created when it was installed. The records to be destroyed must also be disconnected from any sets in which they participate as owners or members. Fortunately, these disconnections are automatically carried out by IDMS at the time when the records are destroyed.

7.3.3 The Control Structure of the Directory Prime Subsystem

Having examined in detail the nature of the functions which the Directory Prime Subsystem is required to carry out I shall now look briefly at its internal design. So far I have established what the subsystem is required to do; I shall now consider how it is to accomplish it.

Our main focus in this section will be an examination of the way in which the total functions of the subsystem are disaggregated into a number of discrete, COBOL program modules. Figure 7.2 illustrates the structure of the system at this module level. Each hexagonal box represents a module, and the connecting lines represent the flow of control between the modules. Thus, the initial entry to the subsystem is via the top module in the diagram, 'M1DIRECTORY'. This calls the module directly below it, 'D1ENTRYDD', which itself calls both 'D2DDPROCESS' and 'DZDBPROCESS', and so on. For simplicity the

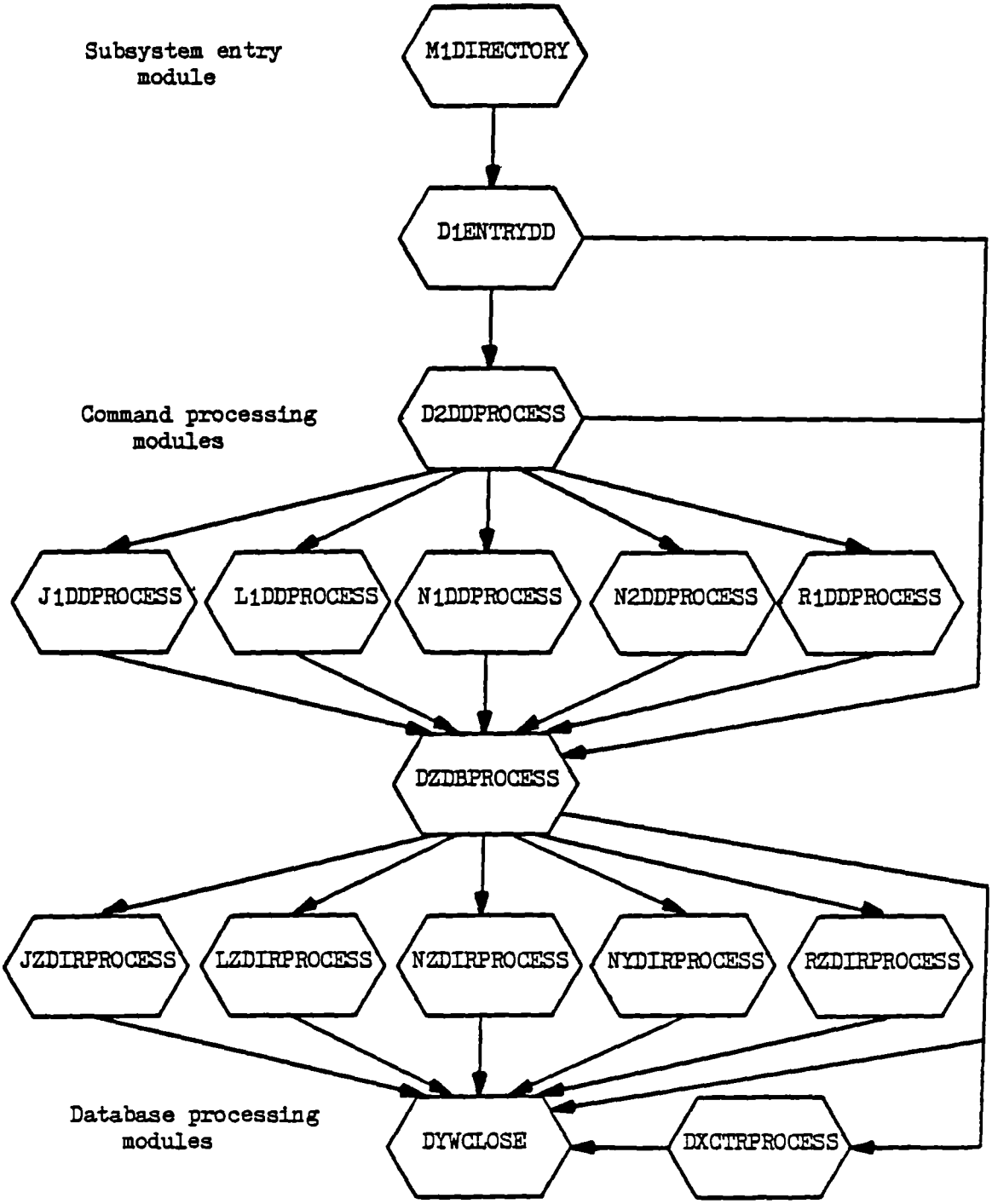


Figure 7.2 The Directory Prime Subsystem: structure and flow of control

interfaces to the Input-Output Subsystem are omitted.

It should be observed how, at a gross level, the overall functions of the subsystem precisely map onto this arrangement of modules. Thus, the eight modules in the upper section of the diagram are concerned either with overall subsystem control or with the handling of the user's command and data input; the eight modules in the lower section of the diagram are concerned exclusively with the organisation and control of the Directory Database. In a similar fashion the arrangement of the two groups of five modules in the upper and lower sections of the diagram reflects the other fundamental, functional property of the subsystem, viz. that it is required to supervise the handling of five independent name directories.

Let us now examine the flow of control through the subsystem and the main functions of each of its modules. The subsystem operates as an independent program, and it can only be entered by a call of the top module in Figure 7.2, 'M1DIRECTORY'. The module directly below this, 'D1ENTRYDD', has responsibility for initiating the opening of input and output files and the database, and correspondingly at the end of the job for the closing of input and output files and the database. It also initiates the printing of opening and closing messages to the user. Module 'D2DDPROCESS' carries out the main semantic checks on the user commands, selects the directory which is to be processed and calls the appropriate command processing module. These modules have responsibilities as follows:

J1DDPROCESS - Job-name Directory

L1DDPROCESS - Placename Directory

N1DDPROCESS - Surname Directory

N2DDPROCESS - Christian Name Directory

R1DDPROCESS - Relationship Name Directory

Each of these modules is responsible for deciding what directory actions are required and submitting appropriate action requests to the main database control module, 'DZDBPROCESS'.

All the modules below and including 'DZDBPROCESS' interface directly with IDMS. 'DZDBPROCESS' is responsible for opening the database and for transferring control, as necessary, to the appropriate action modules. Each of the five modules directly below 'DZDBPROCESS' supervises one of the directories. For example, 'NYDIRPROCESS' supervises the Christian Name Directory, and it has sole responsibility for inserting entries into the directory, subsequently removing them, carrying out an alphabetic traversal of the directory, and so on.

The 'DXCTRPROCESS' module controls the handling of the six 'M-D-ITEM-CNT' records. Thus, for example, when a code value is to be obtained for a new major name which is being inserted into the Christian Name Directory 'DXCTRPROCESS' is responsible for retrieving the appropriate 'M-D-ITEM-CNT' record from the database, extracting the code value, incrementing the value in 'M-D-ITEM-CNT' and writing it back to the database. Finally, module 'DYWCLOSE' arranges to close

the database at the end of a normal run or when an irrecoverable error occurs in some database operation.

It should be clear from this brief analysis of the modular and control structure of the Directory Prime Subsystem that a careful attempt has been made to create a structure which faithfully models the underlying structure of the processes which are being handled. In addition, what should also be apparent is the deliberate attempt which has been made to ensure that each module within the subsystem has the responsibility for handling a discrete and clearly-defined part of the total problem. Both of these aspects of the design are regarded as necessary elements in any strategy for developing computer solutions to complex problems.

7.4 THE INPUT-OUTPUT SUBSYSTEM

Earlier in the chapter, in Section 7.3.1, I referred to the provision of common command and data input facilities by the Input-Output Subsystem, and the way in which the Directory Prime Subsystem makes use of these. It will now be appropriate to examine the design of the Input-Output Subsystem and the way in which these and other facilities are provided.

The main responsibilities of the subsystem were described in Section 6.2. Briefly, it is required to provide facilities for

handling all input and output traffic between each of the other subsystems and the user. In addition to handling command and data input it supervises a variety of output facilities: listings of all user input, error and warning messages, information displays, etc.

The overall module and control structure of the Input-Output Subsystem is quite simple. It is illustrated in Figure 7.3. There are two fundamental ways in which the input-output functions, and the associated modules, subdivide. Firstly, there is the separation into input functions, handled exclusively by the two modules on the left, and output functions, handled exclusively by the other four modules on the right. But, secondly, there is a vertical separation into physical functions and logical functions. Thus, the two modules in the lower section of the diagram are concerned exclusively with the basic, physical handling of the input and output devices, while the four modules in the upper section of the diagram are concerned with the higher level, logical operations, such as unpacking input records and assembling output displays and messages. I shall explore the nature of these subdivisions and the functional properties of the individual modules in the following sections.

7.4.1 The Input Facility

Modules 'F2INPUT' and 'F4INPUT' are responsible for supervising the handling of all command and data input. Module 'F4INPUT' is essentially the device-handling routine. It contains the COBOL i/o

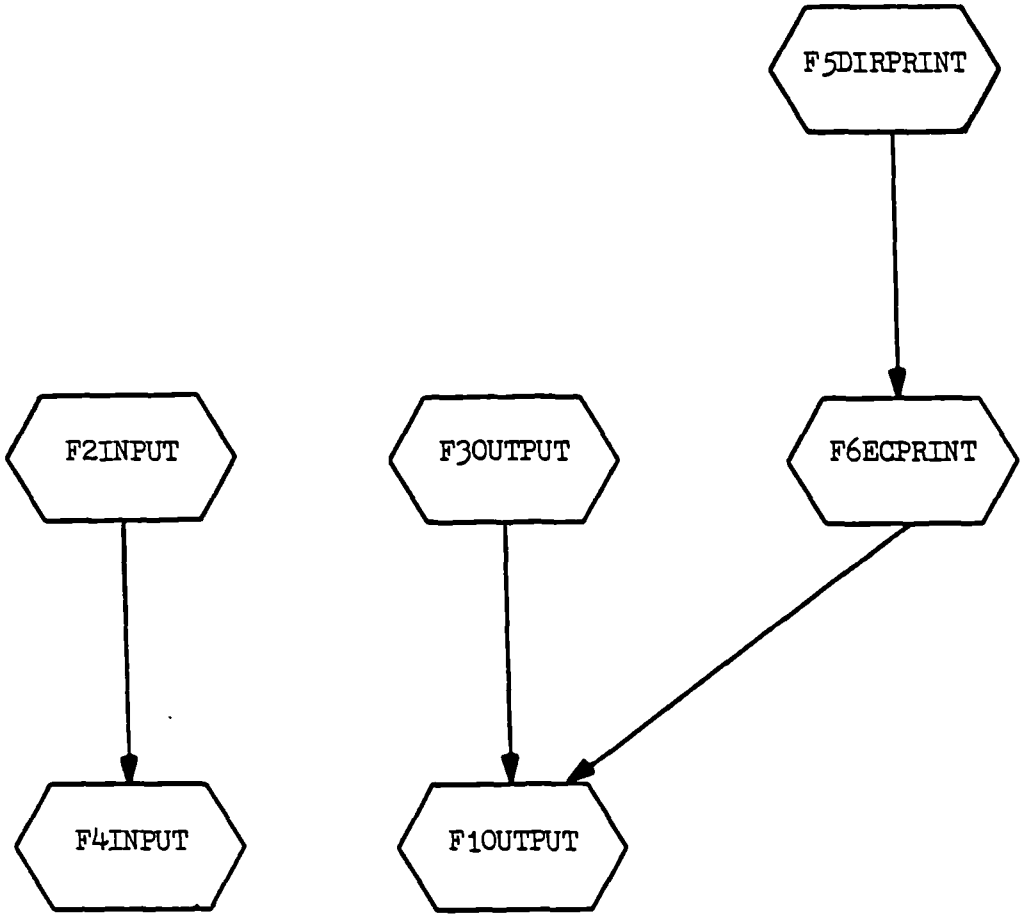


Figure 7.3 The Input-Output Subsystem: structure and flow of control

statements: 'OPEN', 'READ' and 'CLOSE', and its sole functions are to open the input device, read 80-byte card images from it and finally close the device.

Module 'F2INPUT' operates at a functionally higher level, and it makes calls on 'F4INPUT' whenever a physical i/o operation is required, for example, when the next card image is to be obtained. 'F2INPUT' is required to control the handling of the contents of each input record, the following being its main functions:

- the setting up of each new input record in a buffer.

- the initialisation of a special scanning pointer, which can be used to indicate where in the record the processing has reached. When an error occurs this position can be displayed for the user by the use of an asterisk placed under the appropriate character. This will thereby enable him to understand more readily the nature of the error.

- the scanning of the record and the transference of each string, as it is located, into a special string buffer. A caller of 'F2INPUT' is able to specify the range of valid characters which may act as the terminator of a particular string, and 'F2INPUT' will carry out the scanning accordingly. Thus, for example, where the first forename in a name is to be obtained the caller of 'F2INPUT' specifies that a space is the only valid string terminating character. In contrast, where

an occupational description is to be scanned, since this can consist of several words the space character is not specified as a valid terminator. In this case '/', ',' and end-of-record would be specified as valid terminators.

- the skipping of comments within the record. The user may insert comments at any point within a record by bracketing them within the characters '<' and '>', e.g. TAILOR< SAILOR? >. 'F2INPUT' skips over these and does not transfer them to the string buffer.

- the skipping of blank records and the skipping of records after an error. Blank records may be inserted anywhere by the user so as to impose some structure on a file of records: these are skipped over by 'F2INPUT'. Also, after an error is detected in an input record it may sometimes be desirable to skip over the following records up to the next main command record, in order to avoid the production of superfluous 'consequential' errors. 'F2INPUT' provides for this requirement.

- handling continuation records. Where a command or data statement will not fit onto one record the user may continue it onto the following record by placing a 'C' in column 72 and ' /' in columns 1-3 of the following record. 'F2INPUT' is responsible for detecting the continuation characters, reading in the following record and resuming the scanning.

7.4.2 The Output Facility

Modules 'F1OUTPUT' and 'F3OUTPUT' are responsible for supervising the handling of all the main listing output. Their functions precisely parallel those of 'F4INPUT' and 'F2INPUT', in that 'F1OUTPUT' is essentially a device-handling routine and 'F3OUTPUT' operates at a functionally higher level, making calls on 'F1OUTPUT' whenever a physical i/o operation is required.

'F1OUTPUT' contains the COBOL i/o statements: 'OPEN', 'WRITE' and 'CLOSE', and its main functions are to open the output device, write out 132-byte output lines and finally close the device. Additionally, it has facilities for writing out blank lines and advancing the output to the start of a new page.

Module 'F3OUTPUT' contains facilities for formatting output lines in various ways prior to output. The main functions are as follows:

- the output of a 'banner' message at the start and end of each run. For example, at the start and end of a record linkage run it produces the messages 'START OF RECORD LINKAGE' and 'END OF RECORD LINKAGE', respectively. Additionally, at the end of a run it produces a 'SUCCESSFUL RUN' message, or else an indication of the number of warning and/or error messages which have been output.

- the output of warning and error messages. Whenever a module needs to report an error or warning it does this by calling 'F3OUTPUT', specifying the first two characters of the module name, whether an error or warning is to be reported, a number to identify the type of error or warning, and optionally some additional identifying information. 'F3OUTPUT' will then assemble an appropriate message for the user. For example, where the module 'NYDIRPROCESS', within the Directory Prime Subsystem, is requested to insert in the Christian Name Directory a name, say 'ANN', which is already present it will invoke 'F3OUTPUT' to generate a warning message. The message in this case would be as follows (10):

    **** WARNING: NY/01 : ANN

- the output of a row of asterisks for underlining. In Record Linkage and Population Analysis listings it may sometimes be desirable to underline certain names and identifiers for emphasis: 'F3OUTPUT' achieves this by generating the requisite number of asterisks. Thus, for example, the underlining of the name 'WILLIAM BONE' in Figure 6.3 was achieved by the use of this facility.

7.4.3 The Directory Print Facility

The remaining two modules in the Input-Output Subsystem, viz. 'F5DIRPRINT' and 'F6ECPRINT', are provided for use only by the Directory Prime Subsystem. Their sole function is to enable name directories to be displayed in two columns on each listing page. In this form the directories are more convenient for the user to handle and they are also more economical to produce.

Module 'F5DIRPRINT' has responsibility for the detailed formatting of the information which appears in each column, while module 'F6ECPRINT' supervises the arrangement of the material into two columns. Let us now consider briefly the operations of the two modules.

When the Directory Prime Subsystem is instructed to print a name directory it proceeds to send appropriate information, in alphabetic sequence, to 'F5DIRPRINT'. This module is responsible for assembling each single-column print line for the directory and sending it to 'F6ECPRINT': it does not, however, need to take account of the requirement for double-columns.

As 'F6ECPRINT' receives each line of information it does not pass it directly to 'F1OUTPUT' for listing, but, rather, stores it at an appropriate position in a full-page buffer which it maintains. It fills this buffer by putting all the column 1 items in, followed by all the column 2 items. Only when the buffer has been filled in this

way does it then proceed to empty the buffer by sending complete 132-character lines from it to 'F1OUTPUT' for listing.

NOTES

1. This approach reflects the strategy which was adopted during the actual system implementation, i.e. it involved a progressive development of both database and programs in parallel. Thus, for example, at the stage when the Directory Prime Subsystem had been made operational (in November 1977) the database schema contained only the name directories. The design of the Source and Population Database elements, and their incorporation into the schema, was completed later.

2. For illustrative purposes I shall continue to use the code and other attribute values which appear in the corresponding examples in Section 6.1.

3. This particular type of access can, in fact, be viewed as a combination of the first two types of access described. Thus, for example, one can use 'CATHARINE' in association with the first type of access to obtain the code value '000012.02'. One can then derive the code value for the corresponding major name merely by replacing the final two digits by '00', thus getting '000012.00'. Finally, by using this code value in association with the second type of access one would obtain the Christian name 'CATHERINE'.

4. In fact, no such names were presented for insertion in the Christian Name Directory. However, the facility was required for the Placename Directory, for the name '3 CORNERED CLOSE'.

5. Since there is a 1:1 relationship between the 'N-CN-STRING' and 'N-CHRISTIAN-NAME' records one could, in fact, merge together the two record types and their functions. In an alternative design for the Christian Name Directory one might, therefore, choose to remove the 'N-CN-STRING' record and 'N-CN-ENTRY-SET', and renominate the 'N-CHRISTIAN-NAME' record as a CALC-type record. The reason for making use of the two records in the present design is concerned with operational efficiency. By choosing not to nominate the 'N-CHRISTIAN-NAME' record as a CALC-type record one is free to nominate for it the alternative storage mode, 'VIA', and this is specified with respect to its membership in the 'N-CN-ALPHA-SET'. The net effect of using the 'VIA' mode here is that when IDMS is requested to store an N-CHRISTIAN-NAME' record it tries, as far as possible, to store it physically close to the owning 'N-CN-INITIAL' record in the 'N-CN-ALPHA-SET': it might, for example, be able to store it in the same physical storage block. This strategy imposes some physical clustering of the records, which can be used to aid operational efficiency. Records stored using the 'CALC' mechanism, however, tend to be fairly randomly dispersed, and, as such, they cannot obviously enjoy the benefits of physical clustering.

6. At the other extreme the membership of a record in a set can be specified as 'OPTIONAL MANUAL'. This will mean that a record can exist in the database without being a member of that set, and also that the application program must arrange to insert the record into the set using the DML 'CONNECT' instruction. Sets with these membership characteristics are used to model genealogical relationships in the Population Database: the method of organisation will be discussed in Chapter 9.

7. There are, of course, names, such as 'THOMAS' and 'OLIVER', which can occur as genuine Christian names and surnames, and which, as a result, may still require to be entered in both directories.

8. The maximum number of major names which can be stored in each directory is currently set at 99,999. And for each major name there can be a maximum of 99 synonyms. These allocations could be easily increased, merely by employing a longer variable field for each code value.

9. The initial section of a listing of a Christian Name Directory was displayed in Section 6.1.

10. In the present scheme the type of error or warning is uniquely identified by module and condition number codes. In the example shown these are 'NY' and '01', respectively. With the system as currently implemented therefore the user would need to look up these codes in a user manual to find out their significance. However, a suitable enhancement to the system could enable 'F3OUTPUT' to use the codes to access a file of error messages, and so produce a more meaningful message.

CHAPTER 8 IMPLEMENTATION OF THE SOURCE SUBSYSTEMS

In this chapter I shall be concerned with the provision of facilities for handling source records and assembling them in readiness for linkage. My main aim will be to examine the nature of the processes which are needed to transform the source records from the user-oriented form in which they are originally submitted into the system-oriented form in which they will ultimately reside in the Source Database.

The transformations which are needed are essentially of two distinct kinds, as follows:

1. a field-by-field translation of the contents of each record, the objective being to convert each field from the form in which it was entered by the user to a form which will facilitate the processes of record linkage.

2. a structural transformation of the entire record into a form which will enable it to be indexed via any surname which it contains. At the same time there is to be an attempt to transpose the potentially large number of different record input formats into a small number of unified 'internal' formats. For example, in the transformed state there should be a single, unified structure for holding any birth or

baptism record.

I shall examine these two kinds of transformations, in turn, in the two major sections of this chapter.

In Section 8.1 I shall be concerned with source translation facilities. For illustrative purposes I shall focus on the translation of 1871 census records, and shall look closely at the field transformations which are required. I shall then examine the nature of the processes which are involved in source translation, looking, in particular, at a number of the consistency checking procedures which may be used to vet the data. Finally, I shall examine the overall design of the program which is responsible for the execution of the translation processes, viz. the Source Translation Subsystem.

In Section 8.2 I shall be concerned with the facilities which are needed to accomplish the structural transformation of the source data. This will focus on the method of loading the data into a suitably designed Source Database. Initially I shall restrict my attention to the handling of 1871 census data and to the way in which it may be organised in the Source Database. After looking briefly at the organisation of the other types of source data I shall then examine the nature of the processes which are involved in creating the Source Database and loading records into it. Finally, I shall look at the overall design of the program which has responsibility for these loading operations, viz. the Source Loading Subsystem.

8.1 THE SOURCE TRANSLATION SUBSYSTEM

In Chapter 5 the Source Translation Subsystem was used extensively as an example in the discussion of programming and database methods. It should therefore be possible to deal more briefly with it in this chapter.

In Section 6.3 I investigated the functional requirements of the Source Translation Subsystem, and examined the kind of user interface which is needed. Briefly, the objective is that the subsystem should accept source records from the user in a form which is most convenient for him and convert them into a form which is most convenient for subsequent processing by the system. For maximum convenience for the user the input record formats need to be closely modelled on the formats of the original source documents; for maximum convenience for the system the output record formats need to have a simple, fixed structure, with particular kinds of data items (e.g. dates) having a regular, fixed length form of representation.

The system as currently implemented provides facilities for translating census records for the censuses of 1851, '61 and '71, and for translating baptism, marriage and burial records from parish registers. In the following section I shall explore the nature of this process by examining the transformations which occur during the translation of some 1871 census records.

8.1.1 The Translation Process

Figure 8.1 shows the first section of the output listing from a source translation run in which the 1871 census records for Elwick Hall Parish are converted into internal coded form. The original input source records are shown on the left of the listing; the corresponding internal coded forms are shown on the right.

Even a cursory glance at the internal code on the right will reveal that it has a much more regular structure than have the original source records on the left. Each record starts with a 4-character 'header' field, which serves to identify the essential, physical properties of the record. Thus, for example, '48CE' at the start of the first record indicates that it is 48 characters long and that it contains census data. The presence of such a header field at the start of every record facilitates their physical handling in the internal coded file where they are subsequently stored.

The internal coded file which is produced contains three different kinds of records, and the fifth character of each record identifies the type. The character 'B' in the first record identifies it as a block record: the other type codes are 'H' (indicating household) and 'P' (indicating person). (1)

It should be emphasized that this method of organising records, with particular character positions having a predefined significance and range of values is considerably simpler to handle internally than

```
***********        START OF SOURCE TRANSLATION          ***********

   **  START OF C7 CENSUS BLOCK FOR ELWICK HALL PARISH

:SS/CE/C7/ELWICK HALL PARISH                                          48CE B C7 0007200B ELWICK HALL PARISH
                                                                           W 0000
:
:
:C7/H/1/HIGH STOTFOLD                                                 24CE H 000100103008O000 W 00
:C7/P/WILLIAM JACKSON/HEAD/UNM/48//FARMER,ACRES=392,MEN=3,BOYS=2,      C 64CE P 000058001000840OWOM 0000100K U 63096183
:  /WOMEN=2<LABOURERS>/DURHAM,PITTINGTON                                    0002000 0017200NO LOO 0000
:C7/P/MARGARET DO/SISTER/UNM//48//DO,DO                               64CE P 000038001000840OMOF 0000600K U 63096183
                                                                           0000000 0017200NO LRO 0000
:C7/P/JANE STONEHOUSE/SERV/UNM//23/SERVANT/DO,FISHBURN                64CE P 000032001001580OJOF 0002500N U 72227183
                                                                           0005200 0008000NO LOO 0000
:C7/P/THOMAS JOHNSON/SERV/UNM/32//FARM SERVANT<INDOOR>/DO,SHERATON    64CE P 000056001000860OTOM 0002500N U 68940183
                                                                           0002200 0019300PO LOO 0000
:C7/P/ALBION WORTHY/SERV/UNM/17//DO<INDOOR>/DO,BILLINGHAM             64CE P 000003001001740OAOM 0002500N U 74418183
                                                                           0002200 0001700NO LOO 0000
:C7/P/WILLIAM GREY/SERV/UNM/13//DO<INDOOR>/LINCOLNS,GRIMSBY           64CE P 000058001000630OWOM 0002500N U 75879183
                                                                           0002200 0008500NO LOO 0000
:
:C7/H/2/DO                                                            24CE H 000200103008O000 W 00
:C7/P/ROBERT BELL/HEAD/MAR/33//FARMER/DURHAM,BISHOPTON                64CE P 000049001000130OROM 0000100K M 68574183
                                                                           0002000 0002000NO LOO 0000
:C7/P/JANE DO/WIFE/MAR//39//DO,SEDGEFIELD                             64CE P 000032001000130OJOF 0000200K M 66383183
                                                                           0000000 0019200NO LRO 0000
:C7/P/JOHN THOMAS DO/SON//11///DO,CATLEY HILL                         64CE P 000033001000130OJTM 0000300K O 76610183
                                                                           0000000 0003700NO LRO 0000
:C7/P/FREDERICK DO/SON//9///DO,DO                                     64CE P 000024001000130OFOM 0000300K O 77340183
                                                                           0000000 0003700NO LRO 0000
```

Figure 8.1 The translation of 1871 census records into internal coded form by the Source Translation Subsystem

the more free formatted style of record which is initially presented to the Source Translation Subsystem. In particular, programs are able to make immediate access to the contents of any field; with the original source format, where fields are not in fixed positions, special record-scanning facilities must be used.

Let us now examine briefly the significance of the remaining characters of the first record in the internal coded file. These are as follows:

- C7 This specifies the original source type, viz. census data for 1871.

- 0007200B The seven numeric characters are the internal code for 'ELWICK HALL PARISH', as obtained from the Placename Directory, with the 'B' indicating that this is a base zone location.

- ELWICK HALL PARISH This is the only 'external' character form which is held in the file, and it essentially duplicates the information held in the previous field. The reason for retaining it is so that at source loading time it can be extracted directly and displayed in a message to the user, so as to confirm the precise nature of the data being loaded (2). If it were not present then it would be necessary to provide a directory facility for the Source Loading Subsystem just to translate the '0007200' code.

- W 0000 The 'W' indicates that the whole of the parish data is provided (3), and the final four zero digits serve to pad the record out to 48 characters (4).

The first record is an internal coded form of the first source record which was presented for translation. For each subsequent source record there is a corresponding internal coded equivalent. I shall now examine the contents of the second and third internal coded records, one describing the first household and the other the first occupant of this household.

The contents of the second record are as follows:

- 24CE H This indicates that this is a census household record and that it is 24 characters long.

- 000100103008 0000 These characters contain household address information. The first four characters, '0001', identify the household number, as specified in the enumerator's schedule. The following section, '0010300', is the internal code for 'HIGH STOTFOLD', with the additional 'B' once again indicating that this is a base zone location (5). The next four character positions, '0000', are, in fact, not being used here: they provide a storage location for a house number within a street, where one is present.

- W 00 As before, the 'W' indicates that the whole of the household data is provided (6), and the final two zero digits act as padding characters.

The contents of the third record are as follows:

- 64CE P This indicates that this is a census person record and that it is 64 characters long.

- 00005800100084000M This string serves to identify the name and gender of the individual. '00005800' is the code for 'WILLIAM' and '10008400' is the code for 'JACKSON'. The 'W' is the person's first initial: '0' in the next position indicates that there is no second initial. Finally, 'M' indicates the person's gender, which is male in this case.

- 0000100K U This section contains relationship and marital status information (7). '0000100' is the code for 'HEAD', with 'K' indicating a kin relationship (8) and 'U' the fact that the person is unmarried.

- 63096183 This is the person's computed date of birth, as calculated from his age at the time of the census. '63096' represents the number of days which elapsed between the system's reference date, 31 December 1649, and his birthdate, with '183' representing the precision in days. Thus, the computed date of birth specified here is 63096 +- 183 days.

- 0002000 This is the internal code for 'FARMER'.

- 0017200NO This string contains birthplace information. '0017200' is the code for 'PITTINGTON', 'N' indicating that this is a non-base zone location. The final character position, which is not used here and is set to zero, is intended to be used for a birthplace code when dealing with 1841 census data (9). The code would indicate only very broadly where the person was born. For example, it might indicate only that he was born in his present county of residence.

- L00 0000 The first three characters are employed as record 'validity' indicators. Part of the function of source translation is to carry out an exhaustive validation of the input data, and the three indicators in each of the person records provide a summary of the results of this validation. Subsequently, at record linkage time, it is possible to inspect the indicator values and take appropriate action.

  The first indicator, containing the character 'L' in this case, serves to identify this as a linkable person. If during the translation process it is possible to furnish codes for the person's first Christian name, his surname and computed date of birth, then there is considered to be sufficient information to attempt linkage, and so the person is marked as 'linkable'. Unless this code is set the record will be ignored when linkage is subsequently attempted, and a warning

message will be sent to the user.

The second indicator, which is zero in this case, is used to indicate the 'validity' of the relationship for each person in the household who is stated to be related to the head. For example, the second person in the household has an 'R' character in this position, indicating that she is a valid relation. If some checking procedure has thrown doubt on the validity of the relationship, perhaps because of an unreasonable age or name discrepancy, then the relation indicator will be set to zero. At record linkage time inter-person links will be set up only for those who have the 'R' code set.

The final indicator, which is also set to zero in this case, is an illegitimacy indicator. Should there be a household, for example, where the head is unmarried then any son or daughter present will have the character 'I' inserted in this position, indicating illegitimate. The final four zeros are, as before, used for padding.

The source translation example presented in Figure 8.1 illustrates the extensive handling of ditto signs. The location of the second household is specified, via 'DO', to be the same as that of the first. It will be observed, therefore, that the code for 'HIGH STOTFOLD', viz. '0010300', has been copied into the corresponding field in the household record for the second household. In a similar fashion the surname and birthplace codes for the second occupant in the first household, viz. '10008400' and '0017200' respectively, have

been copied from the corresponding fields in the record for the first occupant. Finally, it will be observed that a double copying has taken place for the second and third farm servants in the first household: in each case the occupation code has been set to '0002200'. As a result of such copying processes, therefore, each record now becomes completely 'self-standing', in the sense that each attribute field is, where possible, given its authentic value, and there is no longer any reference to an earlier record. This transformation, like the others, makes the subsequent record handling simpler.

For comparison, Figure 8.2 shows the first section of the output listing from the source translation run in which baptism entries from the Elwick Hall parish register are converted into internal coded form. It will be observed that in this case there are only two kinds of records produced: the initial block record and the individual 'birth' records (10). In these records the fifth character, 'B', indicates birth records: other possible values for the character are 'M', indicating marriages, and 'D', indicating deaths. The other fields in the records, such as names and dates, are constructed in precisely the same way as the corresponding fields in the census records. The adoption of this strategy will ensure that in subsequent processing the fields can be handled by common mechanisms.

In this section I have analysed closely the nature of the transformations which occur as source records are converted into their equivalent, internal coded forms. In the next two sections I shall examine the processes which carry out these conversions.

```
***********        START OF SOURCE TRANSLATION          ***********

    **  START OF B1 BAPTISM BLOCK FOR ELWICK HALL PARISH

:SS/PR/BAP/B1/ELWICK HALL PARISH/ANG/30/EXTRACT                    56PR B ANGBAPB1 0007200B
                                                                        ELWICK HALL PARISH          E 000000
:
:
:B1/14 MAR 1813/HENRY/S/RALPH/ISABELLA/ALCOCK/POPLAR ROW/FARMER    96PR B 0000270010023000HOM 1813 59577030 59607
                                                                        0017500B LO 00004600100230000RO
                                                                        0000290010023000IO 0002000 L 0
:B1/13 APR/ELIZABETH/D/THOMAS/JANE/KAY/PLAINTREE HILL/FARMER       96PR B 0000180010029000EOF 1813 59607030 59637
                                                                        0017300B LO 00005600100290000TO
                                                                        0000320010029000JO 0002000 L 0
:B1/17 JULY/CARTER/S/WILLIAM/MARY/CROWE/NEWTON HANSARD/FARMER      96PR B 1002500010003200COM 1813 59702030 59732
                                                                        0016004B LO 0000580010003200WO
                                                                        0000400010003200MO 0002000 L 0
:B1/5 DEC/THOMAS/S/ROBERT/ELIZABETH/SWALWELL/AMMERSTON HILL/FARMER  96PR B 0000560010032800TOM 1813 59843030 59873
                                                                        0000501B LO 0000490010032800RO
                                                                        0000180010032800EO 0002000 L 0
:
:
:B1/8 MAR 1814/ANN/D/THOMAS/ANN/WILSON/MIDDLE STOTFOLD/FARMER      96PR B 0000070010017200AOF 1814 59936030 59966
                                                                        0014700B LO 00005600100172000TO
                                                                        0000070010017200AO 0002000 L 0
:B1/8 APR/WILLIAM/S/JOHN/MARGARET/DOBING/HOLE HOUSE/BLACKSMITH     96PR B 0000580010026301WOM 1814 59967030 59997
                                                                        0010800B LO 0000330010026301JO
                                                                        0000380010026301MO 0000400 L 0
:B1/5 JUNE/WILLIAM/S/GEORGE/SUSANNA/ROBINSON/AMMERSTON HALL/FARMER 96PR B 0000580010013900WOM 1814 60025030 60055
                                                                        0000401B LO 0000250010013900GO
                                                                        0000550110013900SO 0002000 L 0
```

Figure 8.2 The translation of parish register baptism entries into internal coded form by the Source Translation Subsystem

8.1.2 Conversion of Strings via the Name Directories

The most significant operation which needs to be carried out by the Source Translation Subsystem is the conversion of names, e.g. surnames and placenames, into fixed format internal codes. Since, however, all the information for doing this is maintained in the Directory Database the conversion is a very simple operation to execute: each name string is submitted to the Directory Database, and if it is present the corresponding code value will be returned.

Such an elementary database operation requires only a very limited 'view' of the Directory Database, and so the subschema which needs to be used by the Source Translation Subsystem is considerably simpler than the one which is used by the Directory Prime Subsystem. This subschema, which contains only ten record types and five set types, is illustrated in Figure 8.3. Since the mechanisms which are used to retrieve internal code values from the directories were examined at length in Section 5.3.4, I shall not consider them further.

8.1.3 Conversion of other Fields in the Source Records

The process of source translation, as described in Section 8.1.1, includes a number of field transformations for which the use of a directory is inappropriate. Marital status information, for example, is normally presented in such a limited number of forms that the

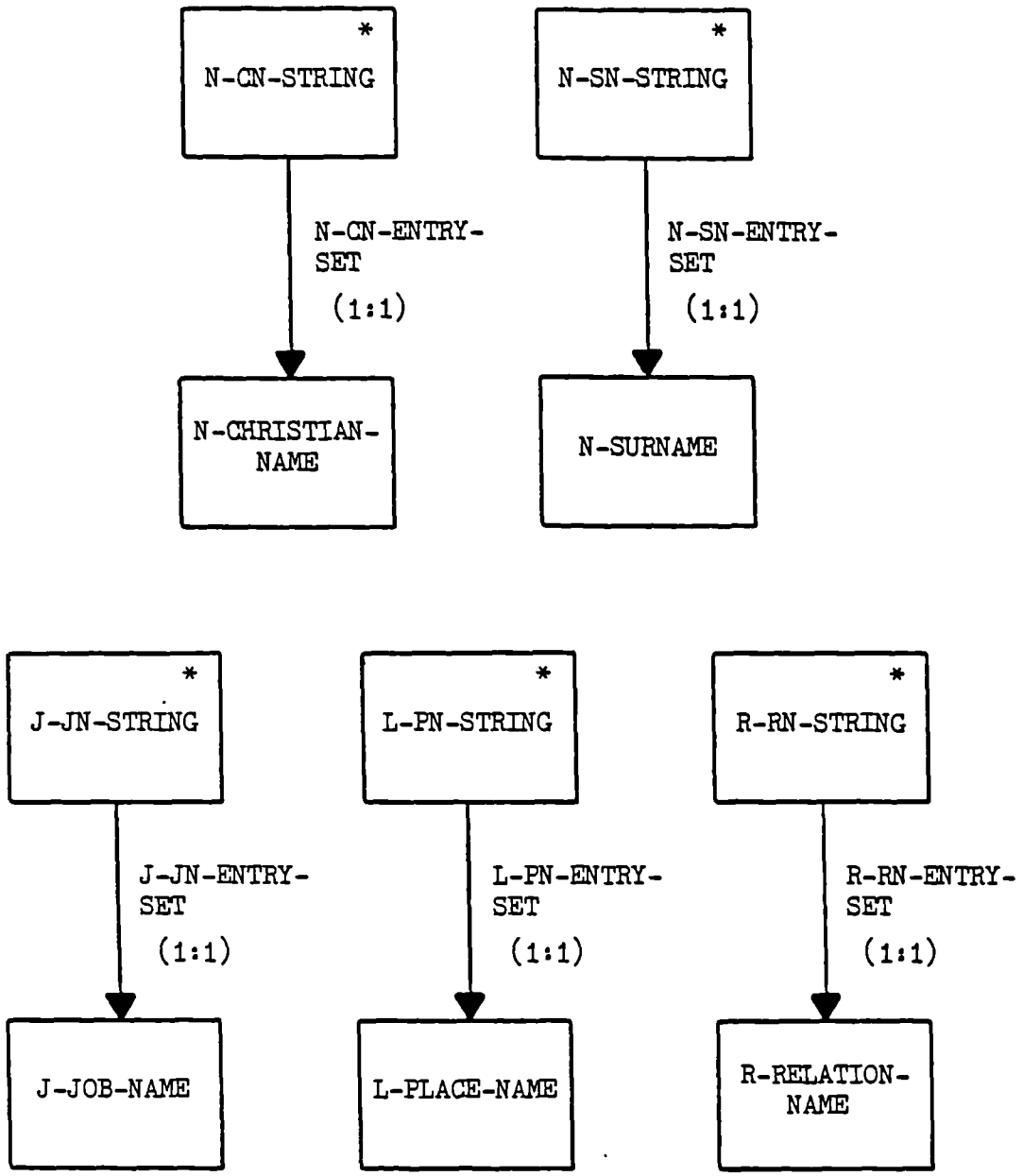


Figure 8.3 The database subschema used by the Source Translation Subsystem
(* = CALC record)

provision of a separate directory to handle it would be hard to justify. Age and date information presents a different kind of conversion problem, but, here again, the employment of a directory does not provide an appropriate solution.

In cases such as these an appropriate solution is to implement special-purpose modules to handle the conversion. Let us consider, as an example, the way in which age information can be handled. There are essentially two discrete operations to be carried out in this conversion. The objective of the first operation is to check the validity of the age information which has been entered. If the validity is confirmed then one can proceed to carry out the second operation. Here the objective is to employ a suitable formula to convert the age into its equivalent, internal coded form, i.e. into computed date of birth form. In the present system design these discrete operations are carried out by independent modules, viz. 'T1STPROCESS' and 'T2TIMEMACHINE', respectively.

8.1.4 Gender Checking

The process of converting individual strings, whether by the use of the name directories or special-purpose modules, can provide a valuable check on the accuracy with which the data has been entered. Thus, for example, if a surname has been mis-spelt as it was entered then it is to be expected that in many cases this will result in a warning message being issued, since the déviant spelling will often

not be present in the Surname Directory (11). Such checks will also be likely to identify another type of error where, for example, a surname has been spelt correctly, but has been entered in the wrong field position. Thus, for example, if a surname string is submitted to the Relationship Name Directory then it will almost certainly not be located, and so, again, a warning message will be issued.

The checks described above will not, however, be able to detect certain other, more complex kinds of data entry errors. For example, this will be the case where the person who has keyed in a census household record has transposed certain fields between one occupant and another, but has nevertheless still placed a valid entry in each field. In this situation a more promising method of error-detection is to view the contents of the record from a more global perspective and attempt to confirm that supposedly related items of information are mutually compatible.

Such 'consistency checking' can be carried out at a number of levels. Thus, for example, for census data it is possible to check that the information which is provided about a group of supposedly related individuals is mutually compatible. I shall explore the use of this kind of strategy in the next section. At a lower level it is possible to check that the separate items of information which are provided about each individual are mutually compatible, and in the remainder of this section I shall examine the role which the gender attribute can play in the implementation of such a strategy.

Many information fields possess a gender attribute. For example, Christian names are normally recognisable as either male or female: thus, 'Thomas' is male and 'Ann' is female. Relationship names are also usually either male or female: thus, 'son' and 'father' are male, while 'daughter' and 'mother' are female. However, there can also be names which have a neutral gender attribute. For example, both the Christian name 'Leslie' and the relationship name 'cousin' can be either male or female.

When presented with several items of information about a particular individual, it is potentially worthwhile, therefore, to check that there are no incompatibilities between the gender attributes of the separate items. In particular, it should be confirmed that there are not simultaneous occurrences of male and female attributes for the same person. Since the present design of the Directory Database allows the user to insert gender attributes into the Christian Name, Job-name and Relationship Name Directories, as described in Section 6.1, it has therefore been a relatively simple matter to implement an appropriate gender-checking mechanism.

Let us consider the nature of this mechanism, in relation to its method of handling census information about an individual. The modules which are involved are 'SCCEPPROCESS', which supervises the processing of a census person record, and 'X2SEXCOMPARE', which contains the checking logic.

When starting to process a census person record, module 'SCCEPPROCESS' calls module 'X2SEXCOMPARE', and this module sets to zero two counters, 'SC-MALE-CTR' and 'SC-FEMALE-CTR'. As the scanning of the record proceeds 'X2SEXCOMPARE' is called and passed gender codes, 'M', 'F' or 'N', as appropriate, depending on the corresponding attributes of the strings which are encountered in the record. As this takes place the module increments the two gender counters to register the occurrences of the 'M' and 'F' codes, and where inconsistencies occur it outputs a warning to the user. Thus, for example, if a record contained the name 'WILLIAM JANE SMITH' and the relationship 'DAUGHTER' then after scanning 'WILLIAM' the 'SC-MALE-CTR' would be incremented to 1. After scanning 'JANE' the 'SC-FEMALE-CTR' would also be incremented to 1, and 'X2SEXCOMPARE' would send a warning message to the user. After scanning 'DAUGHTER' 'SC-FEMALE-CTR' would be incremented to 2, but a second warning would not be issued (12). Finally, when the record has been completely scanned 'X2SEXCOMPARE' is invoked and requested to provide the gender value which was dominant, either 'M', 'F' or 'N'. Thus, in the example given above, where 'SC-MALE-CTR' had the value '1' and 'SC-FEMALE-CTR' had the value '2' it would return the value 'F', and this would then be accepted as the appropriate gender code.

8.1.5 Relationship Checking in Census Households

I shall now explore the characteristics of a consistency checking strategy which is particularly suitable for validating post-1841 census data, and which makes use of relationship information. In Section 8.1.1 reference was made to the checking of relationships in census household records and to the setting up of an 'R' code for those occupants who are deemed to have a 'valid' relationship with the head of the household. In this section I shall look more closely at the nature of the checking procedures which are involved.

The module in the Source Translation Subsystem which carries out the relationship check is 'R3CERELCHECK'. Essentially, it is provided with all the information about the head of the household and all the information about some occupant who is described as being related (13). It then carries out a sequence of checks, in which it assesses the mutual compatibility of the two sets of information. It will, for example, compare the ages and the major surname codes of the head and occupant. If all the tests are completed satisfactorily it reports back that the relationship is valid; otherwise, it reports back that the relationship is invalid. It will also indicate, where appropriate, that it has detected an illegitimate relationship. In the situation where the relationship is declared invalid the consequence will be that at record linkage time the relationship information will be ignored. Nevertheless, it is conceivable that information obtained from some other record or combination of records may authenticate the relationship, and so it may still be correctly

established in the Population Database.

The individual checks which are carried out are as follows:

- The details about the head of the household are first interrogated. If some important code is missing, either that relating to the head's surname, gender, relationship, marital status or birthdate, then the check fails. The check also fails if the head's relationship is given as anything other than 'HEAD' or 'WIFE' or else some non-kin relationship, e.g. 'SERVANT'.

  Other checks at this stage are concerned with the mutual compatibility of the head's relationship, marital status and gender information. If the head's relationship is specified as 'WIFE' then a check is made that the gender is 'FEMALE' and that the marital status is 'MARRIED'. Similarly, if the head's marital status is specified as 'WIDOWER' or 'WIDOW' then a check is made that the gender is 'MALE' or 'FEMALE', respectively.

  Unless the basic attributes describing the head of the household are able to satisfy these initial tests it is considered that it will not be feasible to attempt any further validation of the relationship. If, for example, there should be doubt about the marital status of the head then this could clearly introduce problems when subsequently attempting to check relationships such as 'WIFE', 'SON' and 'DAUGHTER'.

- The details about the relative are interrogated in a similar fashion to confirm that they are sufficient and consistent. Firstly, a check is made that the codes for surname, gender, relationship and birthdate are present.

  There then follows a series of tests which are concerned with the mutual compatibility of the relationship, marital status and gender information. For each relationship a check on the gender is carried out. For example, if the relationship is given as 'SISTER' then the gender should be 'FEMALE'. Where the relationship is given as 'WIFE' then a check is made that the marital status is 'MARRIED'. Similarly, if the relationship is given as 'FATHER-IN-LAW' or 'MOTHER-IN-LAW' then a check is made that the marital status is 'WIDOWER' or 'WIDOW', respectively, or else is 'MARRIED'. Finally, if the marital status is specified as 'WIDOWER' or 'WIDOW' then a check is made that the gender is 'MALE' or 'FEMALE', respectively. Again, it is considered that unless the basic attributes describing the relative can satisfy these initial tests it would not be feasible to attempt any further validation of the relationship.

- If the above tests are successfully completed then the next step in the procedure is to check the mutual compatibility of the information which is provided about the head and the relation. Firstly, a check on the major surname codes is carried out. These codes should be identical when the head is 'MALE', or 'FEMALE' and 'UNMARRIED', and when the relation is

'WIFE', 'SON', 'BROTHER', 'FATHER', or else 'DAUGHTER' or 'SISTER' and 'UNMARRIED'. Next, there is a check that if the relationship is 'WIFE', 'FATHER', 'MOTHER', 'FATHER-IN-LAW' or 'MOTHER-IN-LAW' then there has not been a previous occurrence of this relationship in the household. Finally, if the relationship is specified as 'WIFE' then a check is made that the head is 'MALE' and 'MARRIED'.

- In the next test the relative ages of the head and the relation are compared to ensure that they are compatible. Any father is assumed to be 15 to 75 years older than his child; a mother is assumed to be 15 to 50 years older than her child. A husband may be up to 60 years older than his wife; a wife may be up to 40 years older than her husband. Finally, two siblings are assumed to have ages within 35 years of each other (14).

- If all the above tests have confirmed that the relationship is valid then a final test is carried out to establish its legitimacy. The relationship is assumed to be illegitimate when either the head is 'UNMARRIED' and the relationship is 'SON' or 'DAUGHTER', or else the relation is 'FATHER' or 'MOTHER' and the marital status is 'UNMARRIED'. If either condition is detected then the illegitimacy indicator will be set for the child in the relationship.

8.1.6 The Control Structure of the Source Translation Subsystem

So far in this chapter I have examined the nature of the functions which the Source Translation Subsystem is required to carry out, and have observed, in particular, the facilities which it provides for field transformation and validation. It is necessary now to look at the internal design of the subsystem and the way in which its total functions are disaggregated into program modules. Since, however, I have already examined at length, in Chapter 5, the structure and flow of control through a section of the subsystem (15), I shall now concentrate attention only on those aspects which have not already been covered.

Figure 8.4 illustrates the main structure of the subsystem at this module level, and Figure 8.5 shows some additional input/output and support modules, which are used in the source translation process. For simplicity the interfaces to the Input-Output Subsystem are omitted, as are also the connections between the parish register translation modules ('SFPRBPROCESS' and the three modules which it calls) and the record handling modules ('J2STPROCESS', etc.).

Let us initially observe the way in which, at a gross level, the overall functions of the subsystem map onto the arrangement of modules shown in Figure 8.4. Firstly, the hierarchical arrangement of the three census translation modules ('SACEBPROCESS', 'SBCEHPROCESS' and 'SCCEPPROCESS'), on the left of the diagram, maps precisely onto the corresponding internal structure of the data itself, with its

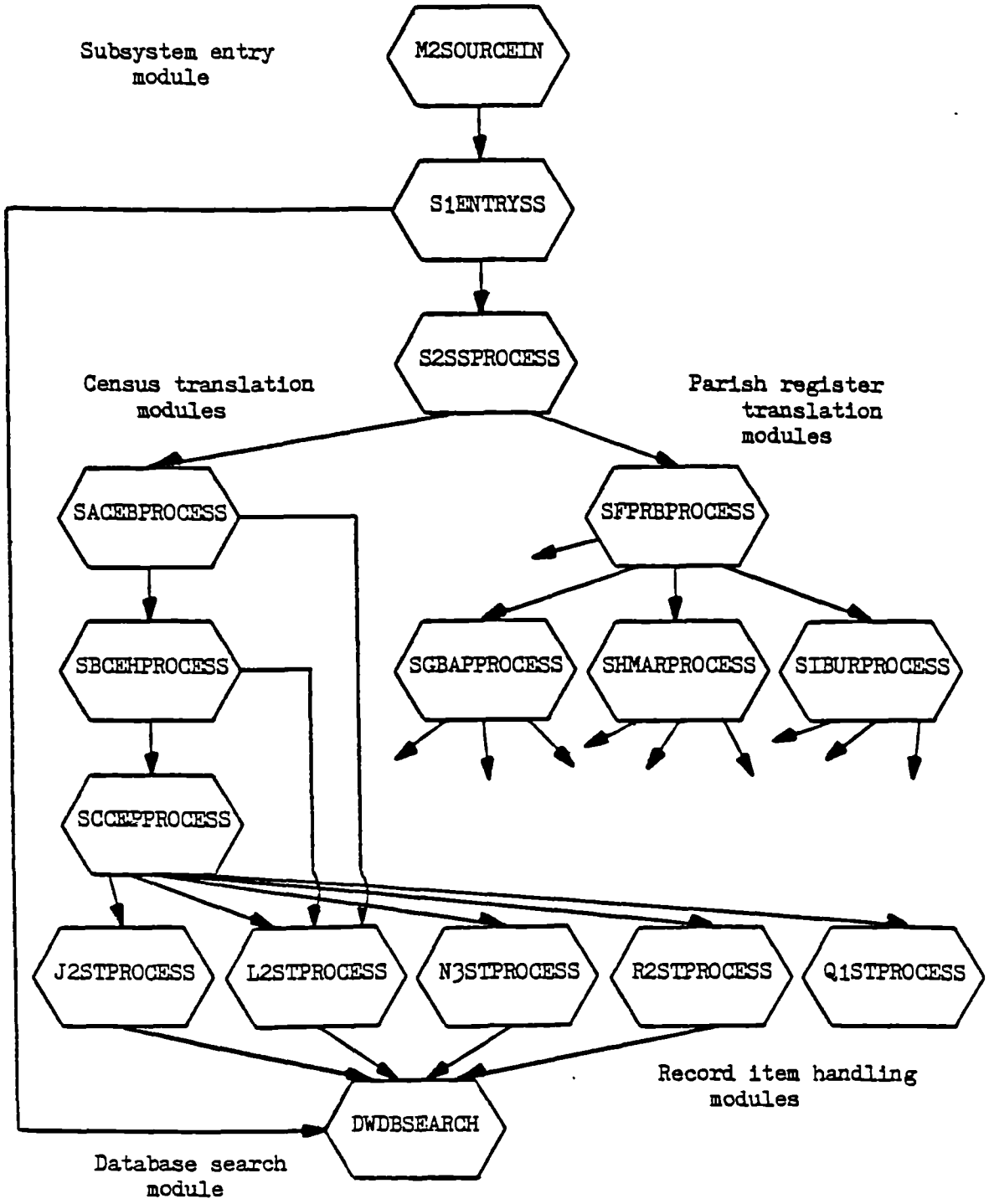


Figure 8.4 The Source Translation Subsystem: main structure and flow of control

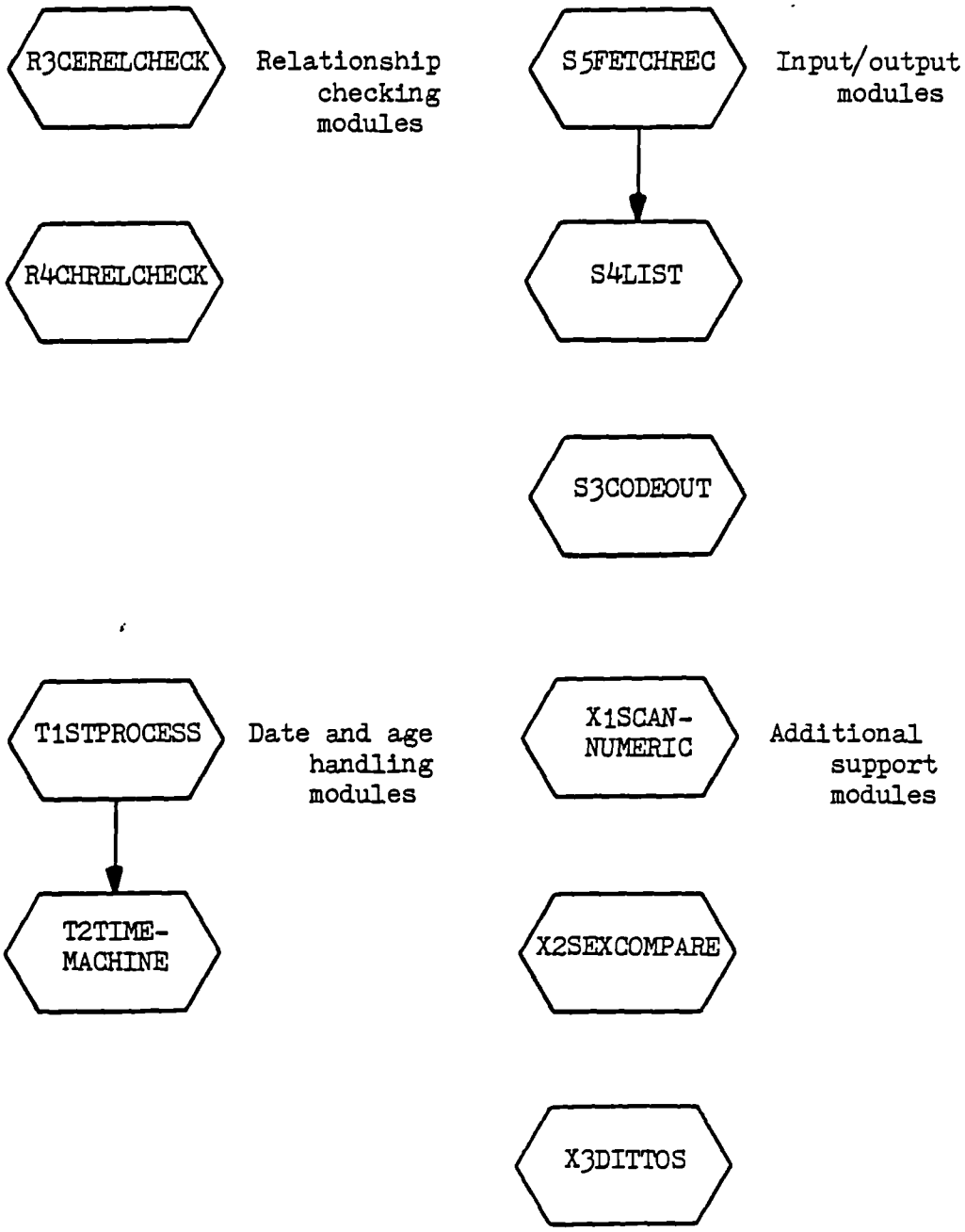


Figure 8.5 The Source Translation Subsystem: input/output and miscellaneous support modules

fundamental hierarchical breakdown into enumeration districts, households and persons (16). The four parish register translation modules, on the right of the diagram, have their own corresponding two-level structure. Essentially, module 'SFPRBPROCESS' is concerned with parish level attributes (e.g. the parish name and denomination) and with the handling of a complete block of records; modules 'SGBAPPROCESS', 'SHMARPROCESS' and 'SIBURPROCESS' are concerned with the translation of the individual baptism, marriage and burial records, respectively.

Most of the other modules in the subsystem are concerned with the translation of the individual items in the source records. The most prominent processing function involves the handling of name information and the associated interface to the Directory Database. Five of the modules in the lower section of Figure 8.4 are involved with the use of the name directories: but, in view of the very limited database access requirements, only one of these, 'DWDBSEARCH', has a direct interface with IDMS.

Let us now examine briefly the flow of control through the subsystem. Entry to the subsystem is via the top module in Figure 8.4, viz. 'M2SOURCEIN'. The module directly below this, 'S1ENTRYSS', has responsibility for initiating the opening of the input and output files and the database, and correspondingly at the end of the job for the closing of input and output files and the database. 'S1ENTRYSS' is also responsible for initiating the printing of opening and closing messages to the user, and for invoking 'S2SSPROCESS' to translate each

block of source records.

The file which is input to a source translation run can consist of records drawn from many different sources, e.g. 1851 census records, parish register baptisms, etc. At the start of each new section of records the user inserts a special block record, containing 'SS' in the first two character positions, which serves to identify the type of data and the record format. The function of module 'S2SSPROCESS' is to check the block record and then transfer control to the appropriate module which will supervise the translation of the source records which follow. Module 'SACEBPROCESS' supervises the translation of a block of census records and module 'SFPRBPROCESS' supervises the translation of a block of parish register records.

Since I have discussed the handling of census records earlier I shall now confine my attention to the handling of parish register records. The function of module 'SFPRBPROCESS' is to generate a block record in internal coded format and then to transfer control to 'SGBAPPROCESS', 'SHMARPROCESS' or 'SIBURPROCESS', depending on whether the block contains baptism, marriage or burial records, respectively. Each of these modules invokes the Input-Output Subsystem to provide new records and to unpack strings from these records. The strings are then presented in turn to the various item translation modules. Thus, for example, all personal names are handled by 'N3STPROCESS', and dates and ages by 'T1STPROCESS'.

Module 'S3CODEOUT', in Figure 8.5, is essentially the device-handling routine for the internal code which is generated. It contains the COBOL i/o statements: 'OPEN', 'WRITE' and 'CLOSE', and its sole functions are to open the output device, write out variable length records to it and finally close the device at the end of the job. As each source-handling module, e.g. 'SGBAPPROCESS', completes the translation of one source record it sends the corresponding internal coded record to 'S3CODEOUT' for output.

In order to produce the translation listings illustrated in Figures 8.1 and 8.2, special output buffering arrangements are needed. As each source record is input it is not immediately listed, but is set up in an output buffer. As the translation proceeds the individual internal coded items are written to the buffer in the high order positions, and only when the buffer is filled is it submitted for output. To provide these buffering arrangements the subsystem has two input/output modules, 'S5FETCHREC' and 'S4LIST', which themselves call the Input-Output Subsystem when physical input and output operations are required.

At the end of a full source translation run the subsystem will have created a sequential file containing a coded version of the primary source records. In the next section I shall consider the nature of the processes which are involved in loading this data into the Source Database.

8.2 THE SOURCE LOADING SUBSYSTEM

It is now necessary to examine the second set of transformations which must be carried out on the source data in order to prepare it finally for linkage. This set of transformations is essentially concerned with the structural organisation of entire records, and in this section I shall be concerned with the two fundamental aspects of this problem:

1. the nature of the transformations, and, more specifically, how the data should be organised in the Source Database.

2. the nature of the transformation processes, i.e. how the data should actually be transferred from a sequential file and restructured in the Source Database.

I shall begin by considering the design of the Source Database.

8.2.1 The Source Database Structure

In Section 6.4 it was argued that nominal record linkage operations could not be conveniently carried out if the nominal records were maintained in sequential files. As a better alternative it was proposed that the records should be stored in a random-access storage device and that appropriate surname-indexing facilities should be provided. In the context of the present system organisation, and

given the availability of the necessary database facilities, it is therefore appropriate that consideration should be given to implementing these record storage facilities in the form of a suitably structured 'Source Database'. The design features of a suitable database structure must now be considered.

There are three main requirements. The first is that the Source Database should be able to hold records of different kinds. At one extreme there are relatively simple records, such as death records, each of which describes a vital event in the life of one individual; and at the other extreme there are more complex records, such as census household records, for which it is necessary to store information of a non-vital kind about a variable number of people. In the latter case, for example, the 1:n relationship which exists between households and occupants would tend to suggest that each household record should be disaggregated and represented in the database by a set in which there would be an owner record to represent the household and a member record for each occupant.

The second requirement is that the Source Database should make use of unified record formats for records of a particular kind. Thus, for example, there should be a single, unified structure for holding any birth or baptism record. At the source input stage it is desirable to provide the user with a range of record input formats, each one modelled on the format of the original source document. But for optimum handling at the record linkage stage it is preferable to replace these by a small number of unified formats.

The final requirement is concerned with the method of accessing the records. It was suggested in Section 6.5 that a user might wish to select for linkage only a small subset of all the records in the Source Database, e.g. just the 1851 and 1861 census records for 'SMITH'. This kind of retrieval would obviously require that the Source Database should possess a surname index, which would enable all the records containing a particular surname to be easily located. And it would also require that for each surname the individual records should be subdivided by their type. In addition, since the more complex records, such as census household records, often contain references to several surnames, it should also be possible for them to be locatable from the index via each of the component surnames.

The record and set types within the schema which are able to satisfy these requirements and provide a suitable Source Database structure are illustrated in Figure 8.6. This structure contains 11 different kinds of records and 14 different kinds of sets. Of the 11 different record types only the two at the top of the diagram do not contain actual source record information. Thus, the 'N-SURNUM' record is used to provide the surname indexing facility, while the 'M-S-ITEM-CNT' record enables unique source record identifiers to be supplied.

In order to explore the database design strategy most simply I shall restrict my attention initially to the organisation of census household data. There are only four record types involved, and I shall consider each of these in turn, as follows:

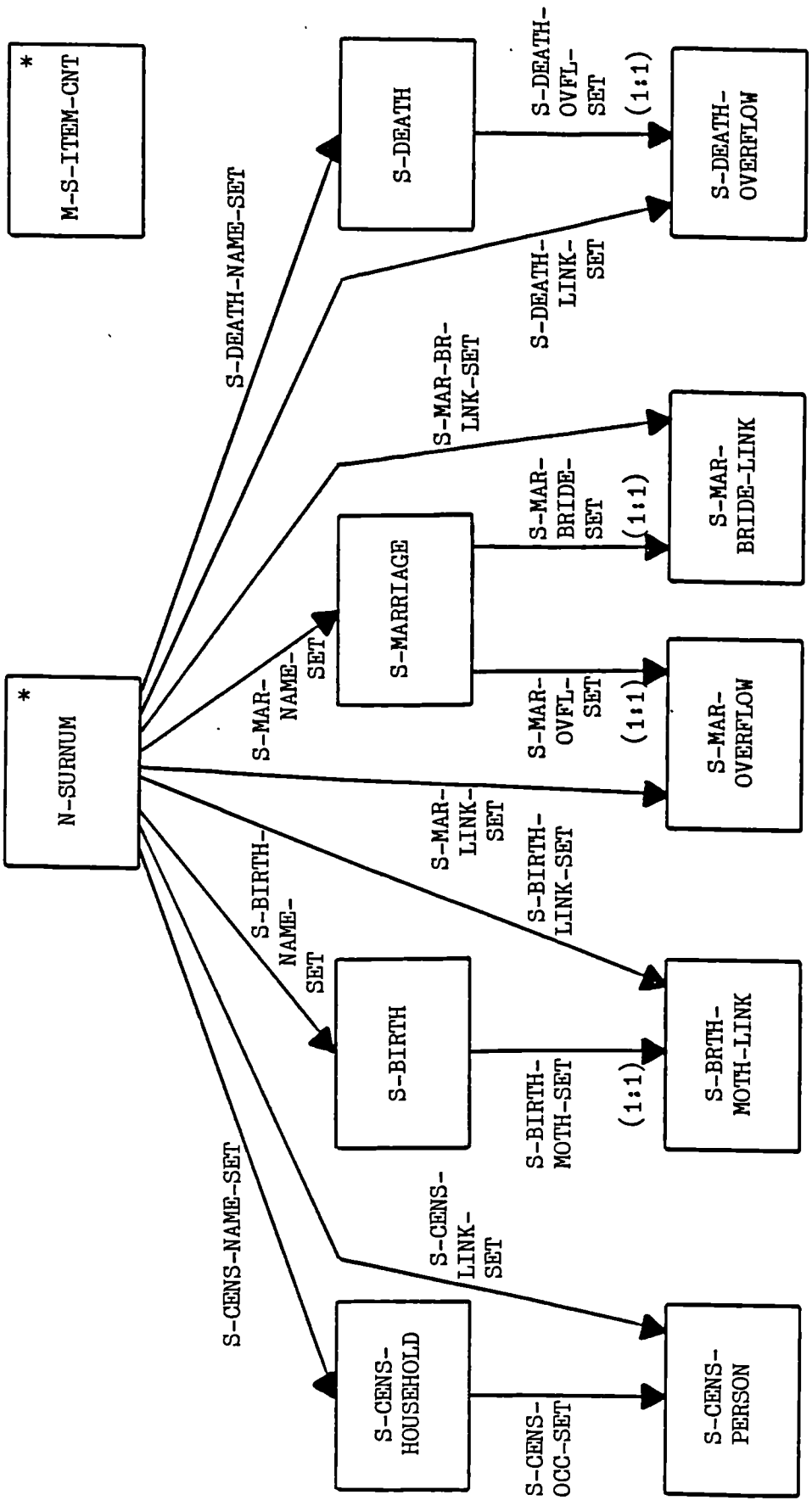


Figure 8.6 A complete set diagram of the Source Database (* = CALC record)

1. M-S-ITEM-CNT. There is a requirement that each source record in the Source Database should have its own unique identifier. In the first place such an identifier can provide the user with a convenient means of cross-referencing from a population analysis listing (as in Figure 6.3) to a listing of the individual records (as in Figure 6.2). Secondly, it can provide the Record Linkage Subsystem with a means for checking which records have been transferred to the Population Database and linked. Details of this checking procedure are provided in Chapter 10.

The 'M-S-ITEM-CNT' record provides the unique values which are allocated to the records: the mechanism is similar to the corresponding one using 'M-D-ITEM-CNT' records in the Directory Database. Within the Source Database there need to be four 'M-S-ITEM-CNT' records, one for each of the main source types: census, birth, marriage and death. When the Source Database is created the four records will be given initial values of '1', and as each source record is loaded the appropriate 'count' value will be used to construct a unique identifier and will then be incremented by '1'. For example, the first census household record to be loaded will be given the identifier 'C00001', the second 'C00002', and so on. Birth, marriage and death record identifiers are created in a similar fashion, except that these have initial characters of 'B', 'M' and 'D', respectively, instead of 'C'.

2. N-SURNUM. As already mentioned, the function of the 'N-SURNUM' record is to provide the surname indexing facility for the Source Database. Given a particular surname it is necessary to be able to locate every source record which contains a reference to that surname. One 'N-SURNUM' CALC record will therefore be created for each surname internal code value, and this record will be the owner in a number of sets in which the different types of source records will participate as members. Having located a particular 'N-SURNUM' record via the 'CALC' mechanism one will then be able to traverse the relevant sets to locate the individual source records. To facilitate this process counter fields are maintained in each 'N-SURNUM' record to indicate how many records are connected in each of the sets.

Thus, for example, to locate all the 1851 and 1861 census records for 'SMITH' we would first of all present the surname string 'SMITH' to the Directory Database, and it would furnish the corresponding internal code, e.g. '10015300'. Next, this internal code would be presented to the Source Database, and the corresponding 'N-SURNUM' record would be found via the 'CALC' mechanism. Finally, if the appropriate counter fields in the 'N-SURNUM' record indicate that census records are connected then the relevant 'SMITH' households and occupants are located by traversing the two sets: 'S-CENS-NAME-SET' and 'S-CENS-LINK-SET'. The precise reason for needing to traverse more than one set will be examined after the

'S-CENS-HOUSEHOLD' and 'S-CENS-PERSON' records have been described.

It should be emphasized that the surname indexing facility which I have described does not have any explicit concept of the clustering of surnames into synonym groups: it is considered to be more appropriate to maintain such 'knowledge' only in the Directory Database. Thus, for example, if for the surname 'SMITH' there is a synonym 'SMYTHE', then each of these will be allocated its own 'N-SURNUM' record and will have its own independent groupings of source records. Should it be required to gain access to source records for 'SMITH' and all its synonyms, then prior access to the Surname Directory will therefore be necessary in order to obtain each of the relevant surname internal code values.

3. S-CENS-HOUSEHOLD. As was proposed above, the information about each household is disaggregated into two types of records: one for the household and the other for each occupant. The 'S-CENS-HOUSEHOLD' record contains information obtained from the corresponding household record generated by the Source Translation Subsystem, together with some additional fields which are included to facilitate subsequent processing. The main components of the record are as follows:

   unique identifier, e.g. 'C00051'

   original source type, e.g. 'C6' (i.e. census data for

1861)

address information

information about the head of the household and his wife (if present)

counters, indicating how many people there are in the household, how many of them are related to the head and how many have the same surname as the head.

Information about the head and his wife is placed in the 'S-CENS-HOUSEHOLD' record to simplify operations at the linkage stage (17). The three counters in 'S-CENS-HOUSEHOLD' are maintained chiefly for Source Database 'housekeeping' purposes.

4. S-CENS-PERSON. This record type holds information about a particular household occupant. The individual components correspond with those generated by the Source Translation Subsystem. Thus, for example, the person's name is present in internal coded form, together with his computed date of birth. In addition to the usual information derived from a census entry there are the three special 'validity' indicators created by the Source Translation Subsystem and subsequently made available to the Record Linkage Subsystem.

In order to explore fully the organisation of census household data I must now consider the ways in which the above records participate in the three set types illustrated in this region of the

Source Database (see Figure 8.6). Each 'S-CENS-HOUSEHOLD' record is the owner in a set, 'S-CENS-OCC-SET', in which the members are the corresponding 'S-CENS-PERSON' records for the people in the household: this set therefore models the explicit household structure.

The other two sets, 'S-CENS-NAME-SET' and 'S-CENS-LINK-SET', provide the access routes for the surname index. In a typical household we will normally expect to find that the predominant surname is that of the head and his family, and that the other members of the household (where present) will share an assortment of different surnames. Thus, for example, in the third household in Figure 6.2 there are eight occupants with the surname 'PATTISON', i.e. the surname of the head, and five with alternative, different surnames. A convenient way of indexing such a household is therefore to associate the name 'PATTISON' with the household: this is done by connecting the 'S-CENS-HOUSEHOLD' record to the 'N-SURNUM' record for 'PATTISON' via the 'S-CENS-NAME-SET'. Thus, all the people in the household called 'PATTISON' will be indirectly linked via their 'S-CENS-HOUSEHOLD' record to the appropriate 'N-SURNUM' record. This method of indexing the head's family as a composite group, rather than individually, is an important design feature, geared to the requirements of family-based record linkage. All that remains is to provide some additional means for linking the remaining members of the household to their appropriate 'N-SURNUM' records. This is achieved by means of the 'S-CENS-LINK-SET'. Thus, if one takes the final member of the household, viz. 'MARY A BONE', she will be linked to the 'N-SURNUM' record for 'BONE' via the 'S-CENS-LINK-SET'. If, therefore, we

consider all the census households which appear in Figure 6.2, we can see that for the surname 'BONE' the corresponding 'N-SURNUM' record will own two member households in its 'S-CENS-NAME-SET' and only one member person in its 'S-CENS-LINK-SET'. It should by now be clear how the use of this network of three records and three sets can provide access to all the households containing people with a given surname.

The access to birth, marriage and death records operates in a similar fashion. Thus, the sets 'S-BIRTH-NAME-SET', 'S-MAR-NAME-SET' and 'S-DEATH-NAME-SET' link the records to the 'N-SURNUM' records for the predominant surname. The remaining sets provide access to the records via alternative, subordinate surnames.

Let us consider birth records. In some baptism records and in almost all birth records the maiden surname of the mother is included. In such cases access to the record via this surname can be provided via the 'S-BIRTH-LINK-SET'. But it should be noted that a special record, 'S-BRTH-MOTH-LINK', is created to provide the link in this case. Where we are dealing with baptism records which do not contain the mother's maiden surname there will be a much simpler arrangement, consisting merely of 'S-BIRTH' records as members of the 'S-BIRTH-NAME-SET'.

The method of organising death records is rather similar. Burial records after 1812 usually contain only the name of the deceased: they are therefore simply stored as 'S-DEATH' records and are linked to the appropriate 'N-SURNUM' record via the 'S-DEATH-NAME-SET'. However, in

some burial registers the records for infant and child deaths contain the names of the parents, and in some cases also the maiden surname of the mother. The provision of the 'S-DEATH-OVERFLOW' record and its associated sets is intended to cope with this additional information. In particular, where the mother's maiden surname is included the 'S-DEATH-OVERFLOW' record can be connected in the 'S-DEATH-LINK-SET', and so access to the record via the maiden surname is achieved.

The most complicated surname indexing arrangements in the Source Database are those concerned with marriage records. Here there are three possible access routes to each record. The 'predominant' route is via the groom's surname. The second is via the bride's pre-marriage surname: this will usually differ from her maiden surname if the marriage is her second or a subsequent one. And if this is not the bride's first marriage then there can be a third access route via the bride's maiden surname (18). In the case of a bride's second or subsequent marriage her maiden name will not usually be included explicitly. However, it can be deduced with reasonable certainty if her father's name is provided.

Before 1837 marriage records usually contained only the names of the bride and groom, together with their residence and marital status. After 1837 age and occupation information was also provided, together with details of the names and occupations of the fathers. The arrangement of 'S-MARRIAGE' and 'S-MAR-OVERFLOW' records is intended to reflect this change in content. The information from a pre-1837 marriage can be stored wholly in an 'S-MARRIAGE' record; for a

post-1837 marriage it is necessary to include the additional information in an 'S-MAR-OVERFLOW' record.

Access to the marriage record via the bride's pre-marriage surname is provided by the 'S-MAR-BRIDE-LINK' record and its associated set, 'S-MAR-BR-LNK-SET'. In the somewhat rare situation where both the bride's pre-marriage and maiden surnames are not the same, and where also the bride's maiden surname can be deduced from her father's name, then a corresponding maiden surname access route is created by connecting the 'S-MAR-OVERFLOW' record in the appropriate 'S-MAR-LINK-SET'.

Having analysed in detail the structural characteristics of the Source Database I shall now consider briefly what is involved in loading a source record, e.g. a census household, into the database. If the census household contains, say, five occupants then it will be necessary to create one 'S-CENS-HOUSEHOLD' record and five 'S-CENS-PERSON' records. It will also be necessary to access the appropriate 'M-S-ITEM-CNT' record to obtain a unique six-character identifier for the record. In addition, it will be necessary to create 'N-SURNUM' records for any surnames which have not been previously encountered.

Finally, having created the records, it will be necessary to arrange for them to be inserted into their appropriate sets. Let us consider the nature of the three sets involved. The 'S-CENS-OCC-SET' operates essentially as a device for connecting together the

disaggregated parts of a single household, and no 'S-CENS-PERSON' record can properly exist in the Source Database without being connected to its associated 'S-CENS-HOUSEHOLD' record. Therefore, it will be appropriate to define this set in the schema as 'MANDATORY AUTOMATIC': as a result IDMS will automatically arrange to make the set connection as each 'S-CENS-PERSON' record is created.

Similarly, since every household must have a 'predominant' surname (which is nominally taken to be that of the first, named occupant) no 'S-CENS-HOUSEHOLD' record can properly exist in the Source Database without being connected to its associated 'N-SURNUM' record. And so, again, the relevant set, 'S-CENS-NAME-SET', can be defined in the schema as 'MANDATORY AUTOMATIC', and IDMS will arrange to make the necessary set connection as each 'S-CENS-HOUSEHOLD' record is created.

By contrast, membership of the 'S-CENS-PERSON' record in the remaining set, 'S-CENS-LINK-SET', is not mandatory, but is required only where a person is encountered whose surname is not the same as the predominant surname of the household. This set will therefore be defined in the schema as 'OPTIONAL MANUAL', the implication being that the application program (in this case, the Source Loading Subsystem) will be responsible for deciding whether a connection is to be made, and, when it is, for issuing the required DML 'CONNECT' instruction.

8.2.2 Accessing the Source Database

I have now completed my analysis of the nature of the structural transformations which must be carried out on the source data in order to prepare it for linkage. I have also examined a scheme for organising the transformed data in a Source Database. I must now address attention to the nature of the processes which are involved in carrying out the transformations and loading the data into the Source Database. The Source Loading Subsystem is that part of the total system which has responsibility for these functions.

In the last chapter the programming strategy employed within the Directory Prime Subsystem was examined by considering a number of fundamental database tasks. I shall now adopt the same approach with respect to the Source Loading Subsystem.

The tasks which I shall consider are the initialisation of the Source Database and the loading into it of a file of 1871 census records which have previously been translated into internal format. Figure 8.7 shows the output listing from such a run. It will be observed that there are only two commands needed: 'SS/DBINIT' and 'SS/DBLOAD'. It will also be observed that the subsystem displays useful, confirmatory information about the nature of the data which it is loading. Thus, it indicates the type of data and the locality, and also, in the case of census data, the number of households and persons (19).

```
********** START OF SOURCE LOAD **********

:SS/DBINIT
:SS/DBLOAD

** START OF C7 CENSUS BLOCK FOR ELWICK HALL PARISH

** END OF C7 CENSUS BLOCK FOR ELWICK HALL PARISH

        NUMBER OF HOUSEHOLDS = 34
        NUMBER OF PERSONS    = 217

********** END OF SOURCE LOAD **********

**** SUCCESSFUL RUN
```

Figure 8.7 The initialisation of the Source Database and the loading into it of a file of 1871 census records by the Source Loading Subsystem

Let us now consider what takes place when the Source Database is initialised. This is a very simple operation, since it involves only the creation of five 'M-S-ITEM-CNT' records. The actions are as follows:

- Firstly a check is made to ensure that the Source Database is a new one. The technique adopted is the same as that used during the initialisation of the Directory Database, i.e. the presence of a special 'M-S-ITEM-CNT' 'marker' record is checked. If it already exists then the initialisation is curtailed and an error message is sent to the user.

- The special 'M-S-ITEM-CNT' marker record is created.

- For each of the four main source types, i.e. census, birth, marriage and death, an 'M-S-ITEM-CNT' record is created, containing the initial value '1'. These are the records which are subsequently used to furnish unique six-character identifiers for the source records as they are loaded.

The actions which take place when a file of 1871 census records is presented for loading are more intricate. The main actions are concerned with the actual process of transferring the census information to the Source Database and with the creation of the required 'S-CENS-HOUSEHOLD' and 'S-CENS-PERSON' records. The other major actions involve the connection of these records to appropriate 'N-SURNUM' records, in order to provide surname indexing. The

detailed actions are as follows:

- The initial block record at the start of the file is read in, and the contents are used to display an informative message to the user, as shown in Figure 8.7.

- The first household record is then read in, followed by the first occupant record. The information from the household record is used to set up corresponding fields in the 'S-CENS-HOUSEHOLD' buffer record in the WORKING-STORAGE SECTION. A six-character identifier for the record is obtained from the appropriate 'M-S-ITEM-CNT' record. The next operation is to present to IDMS the surname code value for the head of the household, with a request that the corresponding 'N-SURNUM' record is to be located. If it does not exist then it must be created. Next, IDMS is requested to transfer the 'S-CENS-HOUSEHOLD' buffer record into the database, i.e. to create the record: it will simultaneously arrange for it to be connected to the appropriate 'N-SURNUM' record in the 'S-CENS-NAME-SET'.

- The information from the first occupant record is used to set up corresponding fields in the 'S-CENS-PERSON' buffer record in the WORKING-STORAGE SECTION. IDMS is then requested to create the 'S-CENS-PERSON' record: while storing this record in the database IDMS will simultaneously arrange for it to be connected to the previously loaded 'S-CENS-HOUSEHOLD' record

in the 'S-CENS-OCC-SET'.

- The second occupant record is read in from the file and the information is used to set up fields in the 'S-CENS-PERSON' buffer record, as for the first occupant. A second 'S-CENS-PERSON' record is then created, and it will automatically be connected into the 'S-CENS-OCC-SET'.

- A check is then made to see whether the surname code value for the second occupant is identical to that of the first. If it is then no further action is needed. If there is a mismatch then it will be necessary to connect the second 'S-CENS-PERSON' into the appropriate 'S-CENS-LINK-SET' i.e. to link the person to the surname index. The occupant's surname code value is used to locate the corresponding 'N-SURNUM' record, or, if it does not exist, to have it created. Finally, an explicit 'CONNECT' instruction must be used to request IDMS to connect the relevant 'S-CENS-PERSON' and 'N-SURNUM' records.

- The remaining occupant records for the first household are read in and are processed in a similar fashion. When the second household record is encountered its first occupant record is also read in and the processing of the second household is then carried out, as for the first. As each 'S-CENS-HOUSEHOLD' and 'S-CENS-PERSON' record is stored in the database, corresponding record counters are incremented.

- When the end of the file is reached an informative message is displayed to the user, indicating how many records have been loaded (20).

8.2.3 The Control Structure of the Source Loading Subsystem

I shall now finally look at the internal design and control structure of the Source Loading Subsystem. This is illustrated in Figure 8.8. Once again, for simplicity, the interfaces to the Input-Output Subsystem have been omitted. In addition, since the three modules at the foot of the diagram are called from most of the other database processing modules the precise connections to them are not shown.

The most significant feature of the internal design of this subsystem is the preponderance of modules which are concerned exclusively with the organisation and control of the Source Database. Thus, of the 12 modules shown in Figure 8.8, all except the two at the top and 'S7CODEIN' interface directly with IDMS. This preponderance of database-handling modules precisely reflects the functional bias of the subsystem: i.e. it has little interaction with the user, but considerable interaction with the Source Database.

Let us now examine the flow of control through the subsystem and the main functions of each of its modules. Entry to the subsystem is via the top module in Figure 8.8, viz. 'M3SOURCELOAD'. The module

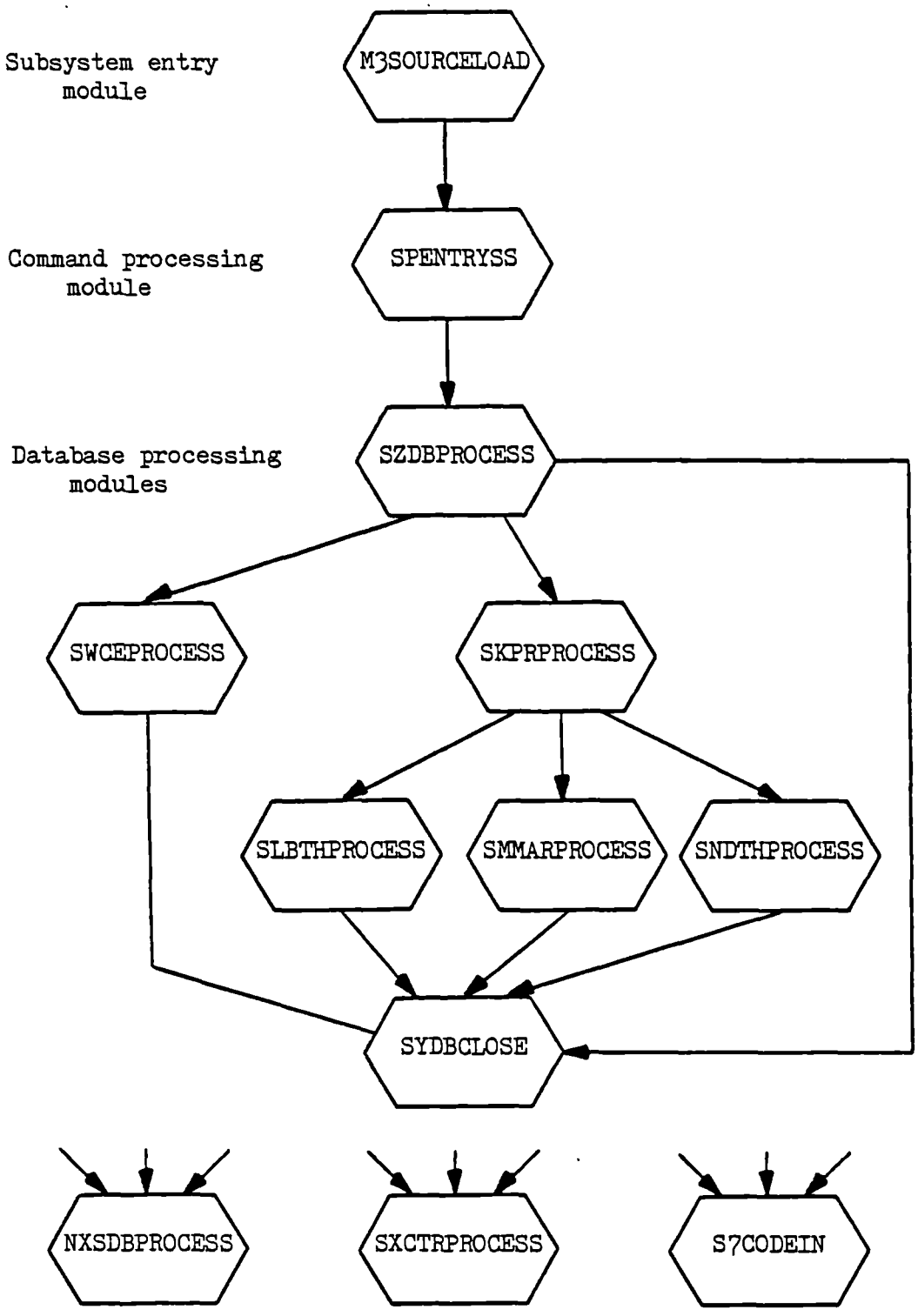


Figure 8.8 The Source Loading Subsystem: structure and flow of control

directly below this, 'SPENTRYSS', has responsibility for initiating the opening of the input and output files and the database, and correspondingly at the end of the job for the closing of input and output files and the database. It is also responsible for initiating the printing of opening and closing messages to the user. Since the subsystem has to deal with only two simple commands, viz. 'SS/DBINIT' and 'SS/DBLOAD', the code for dealing with these is also included in 'SPENTRYSS'. When 'SPENTRYSS' has determined which operation is required it then calls 'SZDBPROCESS' with an appropriate action request.

Module 'S7CODEIN', at the bottom of Figure 8.8, is essentially the device-handling routine for the file of internal coded source data which is presented for loading: its counterpart in the Source Translation Subsystem is module 'S3CODEOUT'. 'S7CODEIN' contains the COBOL i/o statements: 'OPEN', 'READ' and 'CLOSE', and its sole functions are to open the input device, read in variable length records from it, and finally close the device at the end of the job.

Module 'SZDBPROCESS' is responsible for opening the database and for supervising the initialising and loading operations. Thus, for example, at the start and end of a loading operation it invokes module 'S7CODEIN' to open and close, respectively, the source data input file. It also processes each block record in the input file, and depending on the source type it either transfers control to 'SWCEPROCESS' to load census data, or to 'SKPRPROCESS' to load parish register data. From the block record 'SKPRPROCESS' is able to

determine whether the following block of data consists of baptism, marriage or burial records, and so it can transfer control to the appropriate source loading module: 'SLBTHPROCESS', 'SMMARPROCESS' or 'SNDTHPROCESS', respectively.

These three modules and also the census data loading module, 'SWCEPROCESS', obtain their internal coded records via calls on 'S7CODEIN'. When they need to locate or create the surname-indexing 'N-SURNUM' records they achieve this by calling 'NXSDBPROCESS'. And when they need to obtain unique six-character identifiers for their source records they call 'SXCTRPROCESS' to access the appropriate 'M-S-ITEM-CNT' records. Finally, 'SYDBCLOSE' is the database closing module, which is called both at the end of a normal run and also when an irrecoverable error occurs in some database operation.

This description of the modular design of the subsystem should once again serve to illustrate the way in which a subsystem structure can be made to model precisely the underlying structure of the processes which are being handled. At the same time, it should also serve to illustrate the way in which the subsystem structure can be made to map onto the corresponding database structure, with individual modules being given sole responsibility for handling particular record types. These design characteristics are regarded as being of fundamental importance in the disaggregation and solution of complex problems.

I have now completed my examination of the processes which are needed to transform source records in readiness for linkage. In the next chapter I shall begin my analysis of the record linkage processes themselves.

NOTES

1. Spaces are introduced into the listing between major, internal coded items so that they can be more easily identified. Thus, although the first seven characters in the first record are actually '48CEBC7', they are deliberately separated, as shown, to improve the readability.

2. Examples of such messages are illustrated in Figure 8.7.

3. It was envisaged that a researcher might wish to include only certain households from particular enumeration districts. He could, for example, include households for particular family names or those containing one or more natives of the base zone. In view of the possible occurrence of such circumstances it is appropriate, therefore, to include a code which will serve to indicate whether or not the whole of a source document has been submitted.

4. For record storage convenience all records are designed to be some multiple of 8 characters long.

5. It might seem that the inclusion of the 'B' code would be unnecessary here, since the whole enumeration district was described as being within the base zone. However, there are two important reasons for its inclusion. In the first place, there can be locations which are geographically within the base zone, but which have been designated as 'peripheral', and for which the appropriate code will be 'P'. (See Section 3.2.2.) Secondly, when the household records are eventually loaded into the Source Database they are removed from their essentially geographical context and are placed in a 'nominal' context, i.e. so that the records for a particular surname are grouped together. In this situation the zone code within the household record is needed to establish the correct location.

6. It was envisaged that for a very long household, e.g. a workhouse, a researcher might choose to include only certain occupants, such as the immediate relatives of the head of the household. Again, therefore, as for the block record, it is appropriate to include a code which will serve to indicate whether or not the whole of the household has been submitted.

7. In the present implementation only explicitly declared relationships are used, i.e. no attempt is made to deduce the relationships which may exist between those household occupants who are not declared to be related to the head.

8. 'HEAD' is not, of course, a kin relationship, as such. However, by marking it in this way the kin code then clearly distinguishes between those members of the household who are explicitly

described as related to each other (those with the 'K' code) and those who are not so described (those with the 'N' code).

9. Processing facilities for 1841 census data are, however, not included in the present system.

10. The record format, as defined, is intended to be capable of holding information derived from either a birth or a baptism record. If, for example, processing facilities were incorporated for vital registration birth records then the initial 13 characters of the block record might in this case read as follows:

    56VR B GROBTHB5

    where 'VR' indicates vital registration, 'GRO' is an abbreviation for General Record Office, 'BTH' indicates birth and 'B5' identifies a particular record input format.

11. This error-detection capability does, incidentally, provide an additional advantage of employing a name directory for handling surnames, rather than a Soundex-type algorithm. Such an algorithm, as described in Section 3.1.2, is normally designed to accept as valid any alphabetic string which is submitted to it.

12. A warning is, in fact, issued only when a gender value is encountered which conflicts with the net value so far.

13. It carries out the checks, in fact, only for those relationship types which are capable of being subsequently handled by the Record Linkage Subsystem. These are: wife, son, daughter, brother, sister, father, mother, father-in-law and mother-in-law. For all other relationships no relationship is assumed, and the 'R' code is not set up.

14. These demographic checks are broadly in agreement with those described in Wrigley and Schofield 1973, 73-5.

15. The results of this examination were illustrated in Figure 5.2.

16. This structural arrangement was discussed in detail in Section 4.1.3, with particular reference to the adoption of 'top-down' design strategies.

17. This represents a duplication and movement of information 'up' the hierarchical structure, i.e. from 'occupant' to 'household'. Similar duplications and movements of information 'down' a hierarchical structure are also frequently used to simplify the record handling processes.

18. Of course, even this represents a simplification, since it is also possible to index marriage records via the surnames of the witnesses. This provision is, however, not included in the present design.

19. For baptism, marriage and burial data it indicates the number of records loaded.

20. In this example I am considering a file which contains only one block of records. In general, however, a file can contain any number of different kinds of records, and the informative messages will therefore be output at the start and end of loading each block.